Forschungsberichte aus der
Industriellen Informationstechnik | 26

Norbert Mitschke

Konvolutionäre neuronale Netze in der industriellen Bildverarbeitung und Robotik



Norbert Mitschke

Konvolutionäre neuronale Netze in der industriellen Bildverarbeitung und Robotik

Forschungsberichte aus der Industriellen Informationstechnik
Band 26

Institut für Industrielle Informationstechnik
Karlsruher Institut für Technologie
Hrsg. Prof. Dr.-Ing. Michael Heizmann

Eine Übersicht aller bisher in dieser Schriftenreihe erschienenen Bände finden Sie am Ende des Buchs.

Konvolutionäre neuronale Netze in der industriellen Bildverarbeitung und Robotik

von
Norbert Mitschke



Karlsruher Institut für Technologie
Institut für Industrielle Informationstechnik

Konvolutionäre neuronale Netze in der industriellen Bildverarbeitung und Robotik

Zur Erlangung des akademischen Grades eines Doktor-Ingenieurs von der KIT-Fakultät für Elektrotechnik und Informationstechnik des Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation

von Norbert Mitschke, M.Sc.

Tag der mündlichen Prüfung: 22. Oktober 2021
Hauptreferent: Prof. Dr.-Ing. Michael Heizmann, KIT
Korreferent: Prof. Dr. Ralf Mikut, KIT

Impressum



Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

KIT Scientific Publishing is a registered trademark
of Karlsruhe Institute of Technology.
Reprint using the book cover is not allowed.

www.ksp.kit.edu

Print on Demand 2022 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 2190-6629
ISBN 978-3-7315-1197-7
DOI 10.5445/KSP/1000146397

Vorwort

Diese Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für industrielle Informationstechnik (IIIT) am Karlsruher Institut für Technologie (KIT) und wäre ohne die Hilfe und Unterstützung vieler Personen nicht möglich gewesen. An dieser Stelle sei allen gedankt.

Mein besonderer Dank gilt hier Prof. Dr.-Ing. Michael Heizmann nicht nur für die Betreuung dieser Arbeit und die Initiierung der Projekte, an denen ich geforscht habe, sondern auch für die konstruktiven Diskussionen und den beispiellosen Einsatz als Institutsleiter gerade in schwierigen Zeiten. Ebenso möchte ich Prof. Dr. Ralf Mikut für das bekundete Interesse durch die Übernahme des Korreferats danken. Dankbar bin ich auch Prof. Dr. rer. nat. Olaf Dössel, der mich für ein Studium der Elektro- und Informationstechnik in Karlsruhe begeistert hat.

Des Weiteren bedanke ich mich bei den festangestellten Kollegen für die administrative Unterstützung und den zahlreichen Studenten, die durch ihre Abschlussarbeit einen Beitrag zum Entstehen dieser Arbeit geleistet haben. Mein Dank gilt ebenso apl. Prof. Dr. Jörg Matthes, da ohne ihn als externen Betreuer einige Abschlussarbeiten nicht möglich gewesen wären. Außerdem möchte ich mich bei allen wissenschaftlichen Mitarbeitern für die gemeinsame Zeit am IIIT und die Kollegialität bedanken.

Zum Schluss möchte ich mich bei meinen Eltern Martin und Regina und meiner Partnerin Anastasia bedanken, die mich in meinem Tun immer unterstützt haben und mir die Zweifel daran genommen haben.

Speyer, Mai 2022 Norbert Mitschke

Inhaltsverzeichnis

Symbolverzeichnis

Allgemeine Abkürzungen

Abkürzung	**Bedeutung**
bspw.	beispielsweise
bzw.	beziehungsweise
ca.	circa
d. h.	das heißt
engl.	englisch
etc.	et cetera
evtl.	eventuell
ggf.	gegebenenfalls
i. A.	im Allgemeinen
sog.	sogenannt
u. U.	unter Umständen
vgl.	vergleiche
z. B.	zum Beispiel
ADAM	adaptive moment estimation
aSV	automatische Suche nach einer Vorverarbeitungsstrategie
aSVST	aSV-Verfahren mit Straight-Through-Schätzer
BF	Bilaterale Filterung
BN	batch normalization
BS	Bildverschärfung
CCD	charge-coupled device
CLAHE	contrast limited adaptive histogram equalization
CL	s. CLAHE
CMOS	complementary metal-oxide-semiconductor
CNN	convolutional neural network

Abkürzung	Bedeutung
CPU	central processing unit
FPGA	field programmable gate array
GAN	generative adversarial network
GAP	global averaging pooling
GF	Gaußfilterung
GPU	graphical processing unit
gSDK	generalisierter Sørensen-Dice-Koeffizient
H1	Homogenisierung erster Ordnung
H2	Homogenisierung zweiter Ordnung
HE	Histogrammegalisierung
HF	Homomorphe Filterung
HS	Histogrammspreizung
HSV	hue saturation value
i. d. R.	in der Regel
MAC	multiply-accumulate operation
MF	Medianfilterung
MH	Marr-Hildreth-Operator (engl. Laplacian)
MP	maximum pooling
MSig	multiplikative Sigmoid-Aktivierung
MSM	multiplikative Softmax-Aktivierung
ORB	oriented FAST and rotated BRIEF
ReLU	rectifier linear unit
RF	Rechteckfilterung
RGB	rot grün blau
RMSE	root mean square error

Symbole

Lateinische Buchstaben

Symbol	Bedeutung
$a_{\mathrm{MH},i}$	Validierungsgenauigkeit der Topologie $\mathbf{t}_{\mathrm{MH},i}$
$A_{\mathrm{q},\{\mathrm{A},\mathrm{W}\}}$	Amplitude für Aktivierungen und Gewichte
$a_{\mathrm{V},i}$, $\mathbf{a}_{\mathrm{V},i}$	Parameter(-satz) eines Bildverbesserungsverfahrens

Symbol	Bedeutung
$b_{1,\mathrm{A}}$	Breite des Begrenzungsrahmens des Ankers in der normalisierten Bildebene
B	Breite eines Eingabe-/ Ausgabetensors
$\mathcal{B}$	Charge (engl. Batch)
$b_i, b_l, \mathbf{b}$	Biasterm(e) eines Neurons, eines Kernels oder einer CNN-Schicht
$\mathbf{b}_{\mathrm{MH},i}$	Breite der Topologie $\mathbf{t}_{\mathrm{MH},i}$
$\mathbf{C}_{\mathrm{K}}$	Kameramatrix
$\mathbf{C}_{\mathrm{M},i}$	Konfidenz des Segmentierungsmodells
$D^{(i)}$	Tiefe des Eingabetensors bzw. Tiefe der Filterkerne (der i-ten Schicht)
$\mathbf{e}_{\mathrm{m}}$	Fehler des Bildmerkmalsvektors
$\mathrm{f}(\bullet)$	Abbildungsfunktion eines CNN
$\mathrm{f}_{\mathrm{bin}}(\bullet)$	Binarisierungsfunktion
$F_{\mathrm{MH}}(\bullet)$	Fitnessfunktion der Metaheuristik
$F_{\mathrm{V},i}(\bullet)$	Abbildungsfunktion eines Bildverbesserungsverfahrens
$\mathbf{g}_{\mathrm{MH},i}$	Gradient der Topologie $\mathbf{t}_{\mathrm{MH},i}$
$G_{\mathrm{V},i}(\bullet)$	linearisierte Abbildungsfunktion eines Bildverbesserungsverfahrens
$b_{1,\mathrm{A}}$	Höhe des Begrenzungsrahmens des Ankers in der normalisierten Bildebene
H	Höhe eines Eingabe-/ Ausgabetensors
$\mathbf{H}$	Gewichte einer CNN-Schicht
$\mathcal{H}$	Gewichte des CNN
$\mathbf{h}_i$	Gewichte eines einzelnen Neurons
$\mathbf{I}_{\mathrm{aug}}$	augmentiertes RGB-Bild
$\mathbf{I}$	RGB-Bild
$\mathbf{I}_{\mathrm{V}}$	Resultierendes Bild des aSV-Verfahrens
$\mathbf{I}_{\mathrm{H},i}$	RGB-Bild des `dtd`-Datensatzes
$\mathbf{I}_{\mathrm{M},i}$	RGB-Bild des Segmentierungsmodells
$\mathbf{J}$	Jacobi-Matrix
$\mathbf{J}_{\{\mathrm{x},\theta,\mathrm{s},\mathrm{t}\}}$	zu Bildmerkmalen korrespondierende Zeile(n) der Jacobi-Matrix
$J_x, \mathbf{J}_x$	Platzhalter für einen Eintrag in der Jacobi-Matrix
$\mathbf{k}$	Allgemeines Filter

Symbol	**Bedeutung**
$\mathbf{k}_{\mathrm{AA}}$	Anti-Aliasing-Filter
$\mathbf{k}_{\mathrm{Gauss}}$	Gaußfilter
$\mathbf{k}_{\mathrm{Int}}$	Interpolationsfilter
$\mathbf{k}_{l}$	3D-Filterkern einer CNN-Schicht
$\mathbf{k}_{l,d}$	Karte eines 3D-Filterkerns einer CNN-Schicht
$\mathcal{L}$	Zielfunktion des CNN
$L^{(i)}$	Tiefe des Ausgabetensors bzw. Anzahl der Filterkerne (der i-ten Schicht)
$L_{\mathrm{MH},i}$	Tiefe der Topologie $\mathbf{t}_{\mathrm{MH},i}$
$L_{\mathrm{q,F,\{A,W\}}}$	Gleitkommabitlänge für Aktivierungen und Gewichte
$L_{\mathrm{q,I,\{A,W\}}}$	Ganzzahlbitlänge für Aktivierungen und Gewichte
$\mathcal{M}$	Grundwahrheit des Datensatzes
$\mathbf{M}$	Tensor der Grundwahrheit eines Bildes bei der Segmentierung
$\mathbf{m}$	Bildmerkmalsvektor
$\mathbf{M}_{\{\mathrm{A,K,W,R}\}}$	Karte der Grundwahrheit bei der Segmentierung (Anker, Kommutator, Welle, Ritzel)
m_{q}	Momentum bei der Quantisierung
$N_{\mathcal{B}}$	Chargengröße (engl. Batch Size)
N_{chp}	Anzahl der entfernten Filterkerne beim *Channel-Pruning*
$N_{\mathrm{MH},t}$	Populationsgröße der Metaheuristik in der t-ten Iteration
$n_{\mathrm{MH},t}$	Trainingsdurchläufe der Metaheuristik in der t-ten Iteration
N_{q}	Anzahl der Gelenke des Roboterarms
$\mathbf{n}_{\mathrm{q,\{A,W\}}}$	künstliches Quantisierungsrauschen für Aktivierungen und Gewichte
N_{V}	Anzahl der Bildverbesserungsverfahren
$O_{i,\mathrm{AA}}$	Operation i beim AutoAugment-Verfahren
$o_{\mathrm{MH},i}$	Anzahl der MAC-Operationen der Topologie $\mathbf{t}_{\mathrm{MH},i}$
$\mathcal{P}$	Schätzung der Grundwahrheit eines Datensatzes
$\mathcal{P}_{\mathrm{A}}$	Menge der Pixel, die gemäß der Schätzung $\mathbf{P}_{\mathrm{A}}$ zum Anker gehören
$\mathbf{P}$	Schätzung der Grundwahrheit bei der Segmentierung

Symbol	Bedeutung
$\mathbf{P}_{\{\mathrm{A,K,W,R}\}}$	Schätzung einer Karte der Grundwahrheit bei der Segmentierung (Anker, Kommutator, Welle, Ritzel)
$p_{i,\mathrm{AA}}$	Auftrittswahrscheinlichkeit einer Operation beim Auto-Augment-Verfahren
$\mathfrak{P}^{(l)}_{\mathrm{MH},t}$	Population der Metaheuristik in der t-ten Iteration
$q_i, \mathbf{q}$	Stellung von Robotergelenken
$r_{\{20,75,90\}}$	Konvergenzraten
$\mathbf{R}(\boldsymbol{\phi})$	Rotationsmatrix eines Orientierungswinkels
r_{Red}	Reduktionsfaktor beim *Channel-Pruning*
r_{z}	Zoomfaktor
s_{A}	relative Größe des Ankers im Bild
s_{l}	Skalierungsfaktoren bei der Quantisierung
$s_{\mathrm{MH},i}$	Speicherbedarf der Topologie $\mathbf{t}_{\mathrm{MH},i}$
$\mathrm{S}_{\mathrm{M},i}(\bullet,\bullet)$	Funktion zur Berechnung des Ähnlichkeitsmaßes zwischen einem Bild und dem Segmentierungsmodell
$\mathcal{T}_{\mathrm{aug}}$	Augmentierungsstrategie
$\hat{\mathbf{T}}_{\mathrm{M},i}$	affine Transformationsmatrix
$\mathbf{t}_{\mathrm{MH},i}$	topologiebeschreibender Vektor der Metaheuristik
$\tilde{\mathbf{T}}_{i,j}(\bullet)$	Transformationsmatrix der direkten Kinematik
u, v	Position innerhalb des Sensorkoordinatensystems
$u_{\mathrm{A}}, v_{\mathrm{A}}$	Position des Ankers im Bild
u_0, v_0	Hauptpunkt des Sensorkoordinatensystems
$\mathbf{v}, v_{\{\mathrm{x,y,z}\}}$	translatorische Geschwindigkeit
$x^{\mathrm{B}}, y^{\mathrm{B}}$	Position auf der Bildebene
$x^{\mathrm{B}}_1, y^{\mathrm{B}}_1$	Position auf der normalisierten Bildebene
$x^{\mathrm{B}}_{1,\mathrm{A}}, y^{\mathrm{B}}_{1,\mathrm{A}}$	Position des Ankers auf der normalisierten Bildebene
$\mathcal{X}$	Bilddatensatz
$\mathbf{X}$	Eingabetensor einer Schicht des CNN
$\mathbf{X}_d$	Karte des Eingabetensors
$\mathbf{x}_i$	Eingabevektor eines einzelnen Neurons
$\mathbf{x}_{\mathrm{E}}$	Position des Endeffektors
X, Y, Z	Position innerhalb des Welt- bzw. Kamerakoordinatensystems
$\mathbf{Y}$	Ausgabetensor einer Schicht des CNN
$\mathbf{Y}_l$	Karte des Ausgabetensors

Symbol	Bedeutung
y_i	Ausgabe eines einzelnen Neurons
$Z_{\mathrm{B}}^{\mathrm{K}}$	Kammerkonstante der Kamera
$z_{\mathrm{l},\mathcal{B}}$	temporäre Skalierungsfaktoren bei der Quantisierung

Griechische Buchstaben

Symbol	Bedeutung
α_{BS}	Mischfaktor der Bildverschärfung
α_{HF}	Mischfaktor der homomorphen Filterung
α_{M}	Momentum der Konfidenz des Segmentierungsmodells
α_{z}	Momentum beim adaptiven digitalen Zoom
α_{q}	Puffer bei der Quantisierung
$\beta_{l,\mathrm{CP}}, \boldsymbol{\beta}_{\mathrm{CP}}$	Skalierungsparameter beim *Channel-Pruning*
$\delta_{\mathrm{BN},l}$	Biasterm der Chargennormalisierung
$\Delta_{\mathrm{q},\{\mathrm{A},\mathrm{W}\}}$	Breite der Quantisierungsstufe für Aktivierungen und Gewichte
$\Delta_{\mathrm{x}}, \Delta_{\mathrm{y}}$	Pixelbreite und -höhe
ϵ_{CP}	Schwellenwert beim *Channel-Pruning*
Γ	Generalisierungslücke
$\gamma_{\mathrm{BN},l}$	Skalierungsfaktor der Chargennormalisierung
$\gamma_{\mathrm{V},i}, \boldsymbol{\gamma}_{\mathrm{V}}$	Mischfaktor des aSV-Verfahrens
$\lambda_{\{2,\mathrm{q},\mathrm{m}\}}$	Parameter der Zielfunktionen
$\lambda_{\{\mathrm{a},\mathrm{H},\mathrm{s},\mathrm{o}\}}$	Gewichtsfaktoren der Fitnessfunktion
$\lambda_{\mathrm{e}}, \boldsymbol{\lambda}_{\mathrm{e}}$	Schrittweite des bildbasierten Reglers
$\lambda_{\{\mathrm{v},\omega\}\{\mathrm{xy},\mathrm{z}\}}$	Schrittweite des entkoppelten Reglers
$\hat{\mu}_{\mathrm{BN},\mathcal{B},l}$	geschätzter Mittelwert der Chargennormalisierung
$\mu_{i,\mathrm{AA}}$	Stärke einer Operation beim AutoAugment-Verfahren
$\boldsymbol{\omega}, \omega_{\{\mathrm{x},\mathrm{y},\mathrm{z}\}}$	Winkelgeschwindigkeit
$\boldsymbol{\phi}_{\mathrm{E}}$	Orientierung des Endeffektors
$\sigma_{\mathrm{akt}}(\bullet)$	Aktivierungsfunktion einer Schicht eines CNN
$\hat{\sigma}_{\mathrm{BN},\mathcal{B},l}$	Standardabweichung der Chargennormalisierung
σ_{Gauss}	Standardabweichung eines Gaußfilters
$\sigma_{\mathrm{sig}}(\bullet)$	Sigmoid-Aktivierung
$\sigma_{\mathrm{SM}}(\bullet)$	Softmax-Aktivierung
$\theta_{\mathrm{V},i}, \boldsymbol{\theta}_{\mathrm{V}}$	Gewichtungsfaktor des aSV-Verfahrens

Symbol	Bedeutung
$\mathrm{H}_{\mathrm{HM},i}$	Kreuzentropie der Topologie $\mathbf{t}_{\mathrm{MH},i}$
$\nu_{\mathrm{x}}, \nu_{\mathrm{y}}$	Steigung der Bodenebene
ϕ, θ, ψ	Orientierungswinkel innerhalb des Welt- bzw. Kamerakoordinatensystems
ϕ_{A}	Orientierung des Ankers im Bild
$\psi_{\mathrm{A},i}$	Winkel der zu den Ankerkanten orthogonalen Gerade
ρ_{A}	Abstand der Geraden zum Koordinatenursprung
$\rho_{\mathrm{A},i}$	Abstand der Kanten des Ankers zum Koordinatenursprung
θ_{A}	Winkel der zur Orientierung orthogonalen Gerade
ξ_{A}	Korrekturfaktor der relativen Ankergröße

Hochgestellte Indizes

Index	Bedeutung
$(\bullet)^{\mathrm{B}}$	Koordinaten auf der Bildebene
$(\bullet)^{\mathrm{K}}$	Kamerakoordinaten
$(\bullet)^{\max}$	Maximalparameter
$(\bullet)^{+}$	Moore-Penrose-Pseudoinverse
$(\bullet)^{*}$	Parameter im Optimum
$(\bullet)^{\mathrm{T}}$	Transposition einer Matrix
$(\bullet)^{\mathrm{W}}$	Weltkoordinaten

Tiefgestellte Indizes

Index	Bedeutung
$(\bullet)_{\mathrm{L}}$	Lern
$(\bullet)_{\mathrm{obj}}$	Objekt
$(\bullet)_{\mathrm{T}}$	Test

Mathematische Operatoren

Operator	**Bedeutung**
$\lfloor \bullet \rfloor$	Abrundungsfunktion
$\lvert \bullet \rvert$	Betrag bzw. Kardinalität
$(\bullet)_{\downarrow N}$	Dezimierung um den Faktor N
$\odot$	elementweise Matrixmultiplikation
$*$	Faltungsoperator
$\langle \bullet, \bullet \rangle$	Innenprodukt
$(\bullet)_{\uparrow N}$	Interpolation um den Faktor N
$\times$	Kreuzprodukt für dreidimensionale Vektoren
$\lVert \bullet \rVert_2$	Euklidische Vektornorm
$(\dot{\bullet})$	zeitliche Ableitung

1 Einleitung

In den Anfangsjahren des Computerzeitalters hielt sich die Vermutung unter den meisten Ingenieuren, dass die rechnergestützte Bildverarbeitung niemals besser sein könnte als die visuellen Wahrnehmungsfähigkeiten eines Kleinkindes [138]. Seitdem haben sich nicht nur bahnbrechende Verbesserungen in der klassischen Bildverarbeitung ergeben, sondern mit dem Durchbruch von *AlexNet* [92] beim *ImageNet*-Wettbewerb 2012 haben auch konvolutionäre neuronale Netze (engl. *Concolutional Neural Networks*, kurz: CNNs) Eingang in die moderne Bildverarbeitung und darüber hinaus gefunden: Moderne CNNs sind nicht nur Stand der Technik in bildbasierten Klassifikations-, Detektions- sowie Segmentierungsaufgaben und übertreffen damit selbst die menschliche Wahrnehmung [38], sondern werden u. a. auch für die Bildgenerierung, Ausreißererkennung, Steuerung von Robotern bzw. von Charakteren in Videospielen, Analyse von Spektrogrammen und der Komprimierung eingesetzt. Der Sieg von neuronalen Netzen über die weltbesten Go-, Schach- und Shōgi-Spieler 2016 und 2017 [143, 144] haben das immense Potential der neuronalen Netze einer breiten Masse veranschaulicht.

Für Bildverarbeitungsaufgaben auf den gängigen großen Vergleichsdatensätzen haben sich in der Vergangenheit immer komplexere und damit ressourcenhungrigere CNNs als vorteilhaft erwiesen (s. Abb. 1.1). Um eine Überanpassung durch die steigende Zahl an Parametern zu vermeiden, wurden gleichzeitig immer komplexere Regularisierungsmethoden verwendet. Die Verbesserung bzw. die Verminderung der Kosten der Trainingshardware, aber auch Cloud-Dienste für das Training haben ebenfalls zu diesem Trend beigetragen. Dies bremst gleichzeitig die Forschung an kleineren neuronalen Netzen, da diese auf den Vergleichsdatensätzen nicht mit den großen Netzen mithalten können und daher stets eine gute Begründung wie die Echtzeitfähigkeit oder die Implementierung auf spezieller Hardware für deren Verwendung erforderlich ist. Eine Forschungslücke, die sich hier auftut, ist das Anwenden moderner

Regularisierungsmethoden auf kleine CNNs. Wie Abb. 1.1 zeigt, kann eine gute Augmentierung die Genauigkeit von *PyramidNet* deutlich verbessern [30]. Zwar ist der erreichte Testfehler geringfügig höher als der des *AmoebaNet-B*, dafür ist jedoch die Anzahl der benötigten Parameter um mehr als eine Größenordnung kleiner.

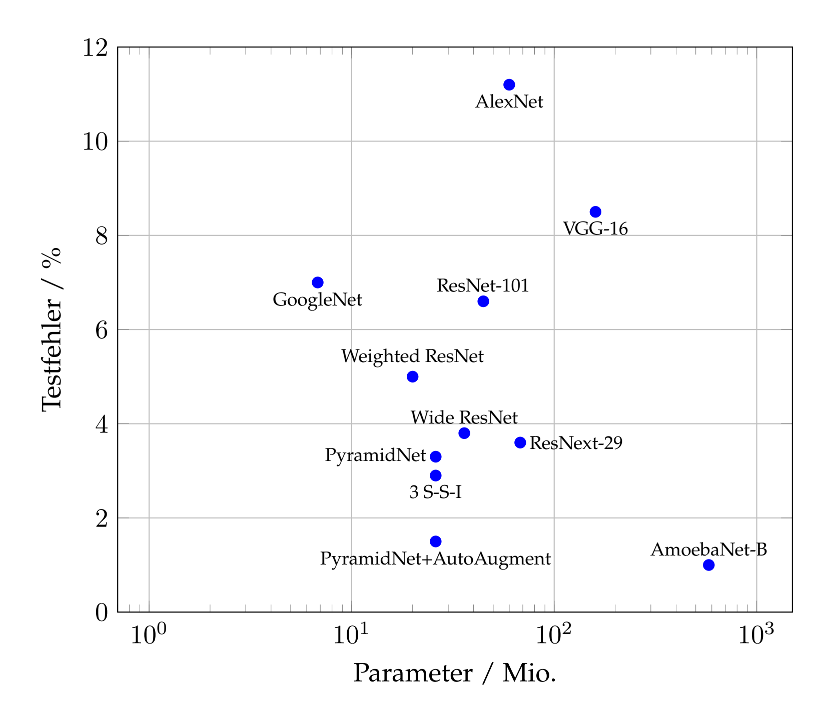


Abbildung 1.1 Erzielter Testfehler auf dem CIFAR-10-Datensatz abhängig von der Anzahl der Parameter für verschiedene CNN-Topologien [30, 48, 63, 92, 104, 127, 131, 139, 145, 155, 172].

CNNs gerieren sich außerdem als *Black-Box*-Modelle, deren Merkmalsextraktoren aus den intrinsischen Eigenschaften eines Lerndatensatzes bzw. beim Verstärkungslernen aus der spezifischen Aufgabenstellung abgeleitet werden. Dies führt dazu, dass sich trainierte Netze leicht stören lassen [53, 73, 119] bzw. nicht invariant bzgl. bestimmter Bildeigenschaften (z. B. Kontrast oder Objektgröße) oder abhängig von der Bildqualität sind [37, 38, 67]. Insbesondere in industriellen oder sicherheitskritischen

Anwendungen ist dies problematisch. Eine geeignete Regularisierung in Form von Augmentierung oder Bildvorverarbeitung hat das Potential, diesbezüglich zu einer höheren Resilienz zu führen und CNNs damit ein Stück weit interpretierbarer zu machen.

1.1 Problemstellung

Mit dem Ziel, künstliche neuronale Netze für industrielle Anwendungen einzusetzen, ergeben sich vielfältige offene Problemstellungen, da wie eingangs beschrieben der Hauptfokus in der Forschung bzgl. neuronaler Netze in der Verarbeitung anderer Datensätze liegt und die Leistung neuronale Netze durch immer komplexere und ressourcenhungrigere Strukturen erhöht wird. Industrielle Bildverarbeitungsaufgaben wie bspw. Texturanalyse, Defektdetektion oder Zeichenerkennung besitzen im Gegensatz zu den häufig verwendeten Vergleichsdatensätzen wie z. B. *ImageNet* oft eine geringere Komplexität, da die Domäne, die der Datensatz abbildet, i. d. R. deutlich kleiner ist: Die Variabilität von Defekten auf metallischen Oberflächen ist bspw. erheblich geringer als die Variationen, die bei der Klassifikation alltäglicher Objekte auftreten. Daher steht in dieser Arbeit die Effizienz der neuronalen Netze im Vordergrund und somit die Frage, wie klein und ressourcenschonend die Inferenz eines CNN sein kann, ohne die Genauigkeit stark negativ zu beeinflussen. Zwar stehen industrielle Anwendungen im Vordergrund dieser Arbeit, jedoch werden die Methoden allgemein entworfen, um auf andere Aufgaben adaptierbar zu sein. Daher werden in dieser Arbeit auch Vergleichsdatensätze aus anderen Domänen untersucht.

Eine weitere Fragestellung, die im zweiten Teil dieser Arbeit untersucht wird, ist die Erkennung und Segmentierung von Objekten unterschiedlicher Bauform bzw. von unterschiedlichem Zustand für das *Remanufacturing*, dem Wiederaufbereiten von Gebrauchtprodukten.

Insgesamt haben sich die folgenden Herausforderungen für den Einsatz neuronaler Netze in der industriellen Bildverarbeitung herauskristallisiert:

- Kleine Datensätze: Obwohl industrielle Aufgaben einfacher sind, mangelt es oftmals an einer ausreichenden Menge an annotierten Bilddaten, da das Annotieren sehr zeit- und kostenintensiv

ist. Daher müssen die vorhandenen Daten bestmöglich genutzt werden.

- Ressourceneffizienz: Das Ziel soll sein, dass die Inferenz des neuronalen Netzes auf einem FGPA implementierbar ist. Hierbei soll der Block-RAM für die Speicherung der Gewichte und Aktivierungen verwendet werden. Dieser hat bei modernen FPGAs der `Xilinx7`-Serie eine Größe von bis zu 7 MB.
- Erkennung von Bauteilen unabhängig von Bauform und Zustand: Dies ermöglicht die Positionierung einer Kamera bzw. das Abfahren einer Trajektorie zum Abscannen der Bauteiloberfläche und anschließender Inspektion.
- *Black-Box*-Modell: Ein neuronales Netz stellt ein *Black-Box*-Modell dar, da die gelernten Merkmale aus statistischen Eigenschaften des Lerndatensatzes abgeleitet bzw. aggregiert werden, was die Erklärbarkeit der Ausgabe eines neuronalen Netzes durch den Anwender besonders erschwert. Daher ist zumindest die Interpretation der Auswirkungen der vorgestellten Methoden mittels geeigneter Theorien erforderlich.

1.2 Eigener Beitrag

Bestehende Ansätze untersuchen entweder die Recheneffizienz oder die Aufbereitung der Datensätze und betrachten daher nicht die Gesamtproblemstellung. Insbesondere den Modifikationen für die Implementierung der Inferenz auf mobilen Geräten und FPGAs kommt weniger Aufmerksamkeit zu, da die Chip-Hersteller einerseits FPGA-Platinen um CNN-Recheneinheiten erweitern oder die Entwickler auf *Cloud*-basierte Lösungen setzen. Beides führt zu einer Verteuerung und Verlangsamung der Inferenz, was den Einsatz neuronaler Netze für industrielle Anwendungen gefährdet oder eine Echtzeitdetektion erschwert. Daher wird in dieser Arbeit ein ganzheitlicher Ansatz gewählt, der folgende Punkte enthält:

- Diskussion des Trainings: In der Literatur finden sich unterschiedliche Ansätze für das Training eines CNN. Allen gemeinsam ist

die Unterteilung eines Durchlaufs (engl. *epoch*) in Chargen (engl. *batches*). Üblicherweise ergibt sich aus der Chargengröße und der Größe des Lerndatensatzes die Anzahl der Iterationen pro Durchlauf, was die Chargengröße in der Literatur zu einem intensiv untersuchten Hyperparameter gemacht hat. In dieser Arbeit werden alternative Definitionen eines Durchlaufs untersucht, da die oben beschriebene Definition bei Online-Augmentierung diskussionswürdig ist. Dies führt zu einer konsistenten Verbesserung beim Training.

- Interpretation von Vorverarbeitung und Augmentierung als Konstruktion invarianter Merkmale: Hierfür wird ein Verfahren für die Bildverbesserung eines Datensatzes entwickelt, mit dem die Daten bzgl. bestimmter Invarianzen in einem Vorverarbeitungsschritt normalisiert werden. Zudem werden mit einem leicht modifizierten *AutoAugment*-Verfahren Invarianzen integrativ durch das CNN erlernt. Beide Verfahren können als Weg für die Konstruktion invarianter Merkmale nach Schulz-Mirbach [137] interpretiert werden. Ferner werden die Wechselwirkungen von integrativer (Augmentierung) und normalisierender (Bildverbesserung) Konstruktion der invarianten Merkmale sowie deren Implikation auf den Ressourcenbedarf und die Resilienz gegenüber Angriffen auf das CNN analysiert. Gegenüber der ausschließlichen Verwendung des *faster-AutoAugment*-Verfahrens nach [61] kann durch die Modifikation und Kombination mit Vorverarbeitung eine deutliche Steigerung der Genauigkeit und Reduktion des Ressourcenbedarfs erzielt werden.

- Quantisierungs-Framework: Für eine sinnvolle FPGA-Implementierung ist eine 8-Bit-Festkomma-Quantisierung notwendig. Hierfür wird ein Quantisierungs-Framework vorgestellt. Dabei werden Quantisierungseffekte schon während des Trainings berücksichtigt und die Verwendung einer Normalisierungsschicht wird vermieden. Durch Erweiterungen und Vorverarbeitung sowie Augmentierung können die erzielten Genauigkeiten weiter erhöht werden.

- Heuristik für die Topologie: Um die Rahmenbedingungen für den Speicherbedarf eines CNN einzuhalten, wird in dieser Arbeit eine neuartige Heuristik beschrieben, die die Topologie eines geeigneten CNN bestimmt. Je nach Wahl der Fitness-Funktion bzw. deren Straftermen können sehr ressourcensparende CNNs gefunden werden. Hierbei muss zwischen Ressourcenbedarf und erzielter Genauigkeit abgewogen werden.
- Bildbasierter Regler mittels U-Net-Ausgabe: Im zweiten Teil dieser Arbeit wird ein bildbasierter Regler entworfen, der auf einer U-Net-Ausgabe basiert. Hierzu wird ein relativ kleiner Datensatz erstellt, mit dem ein leicht modifiziertes U-Net trainiert wird. Das U-Net liefert genaue Schätzungen über die Orientierung, Größe und Lage des Objektes, die sich als Merkmale für den bildbasierten Regler eignen. Dieser ist durch verschiedene Erweiterungen robust gegenüber Rauschen und Fremdobjekten.

1.3 Struktur der Arbeit

In Kapitel 2 werden Grundlagen künstlicher neuronaler Netze und der Robotik beschrieben. Hierzu wird einerseits ausgehend von der digitalen Bildverarbeitung in Abschnitt 2.1 der Aufbau von konvolutionären neuronalen Netzen in Abschnitt 2.2 beschrieben. Anschließend werden in Abschnitt 2.3 die Grundlagen der Robotik geschildert, die für das Verständnis dieser Arbeit notwendig sind. Danach teilt sich diese Arbeit thematisch in zwei Teile.

In den Kapiteln 3 und 4 werden die Verfahren und deren Ergebnisse für den Entwurf effizienter neuronaler Netze beschrieben. Im methodischen Part wird in Abschnitt 3.1 das Training diskutiert. Anschließend werden die Verfahren für die Konstruktion invarianter Merkmale vorgestellt. Diese sind ein Bildverbesserungsverfahren für die Vorverarbeitung in Abschnitt 3.2 und ein modifiziertes Augmentierungsverfahren in Abschnitt 3.3. Anschließend wird in Abschnitt 3.4 die Quantisierung thematisiert, bevor in Abschnitt 3.5 eine Heuristik für den Entwurf neuronaler Netze vorgestellt wird. In Abschnitt 3.6 wird aufgezeigt, wie die einzelnen Verfahren für eine holistische Verarbeitungskaskade kombiniert

werden können. Die Evaluation dieser Verfahren ist Thema von Kapitel 4. Nach der Beschreibung der Simulationsumgebung in Abschnitt 4.1 werden das Training in Abschnitt 4.2 und die Verfahren für invariante Merkmale in Abschnitt 4.3 ausgewertet. Im Anschluss werden das Quantisierungsverfahren in Abschnitt 4.4 und die Heuristik eingebettet im Gesamtsystem in Abschnitt 4.5 untersucht.

Der zweite Teil dieser Arbeit beschäftigt sich mit der robotergestützen Inspektion. Hierfür wird in Abschnitt 5.1 ein neuronales Netz für die Segmentierung von Ankern von Elektromotoren entworfen, auf dessen Grundlage der bildbasierte Regler aus Abschnitt 5.2 fußt. In Abschnitt 6.1 wird erst das U-Net und dann in Abschnitt 6.2 der Regler evaluiert. Hierzu wird in Abschnitt 6.3 ein Fazit sowie ein kurzer Ausblick für zukünftige Arbeiten gegeben.

In Kapitel 7 wird für die gesamte Arbeit eine kurze Zusammenfassung sowie ein Ausblick gegeben.

2 Grundlagen

In diesem Kapitel werden die Grundlagen, die für das Verständnis der Kapitel 3 – 6 notwendig sind, dargestellt. Dieses Kapitel dient einerseits der Einordnung von CNNs in das Feld der klassischen Signal- und Bildverarbeitung und gibt andererseits einen kurzen Abriss des Robotersystems mit frei bewegbarer Kamera, das in den Kapiteln 5 und 6 verwendet wird.

2.1 Digitale Bildverarbeitung

Die rasante Entwicklung von Bildsensorik und Computertechnologie führte dazu, dass mittlerweile in fast allen wissenschaftlichen und technologischen Disziplinen Bildverarbeitung eingesetzt wird. Moderne CMOS- und CCD-Sensoren wandeln hierfür sichtbares Licht durch den photoelektrischen Effekt in Ladungen um. Die angesammelte Ladung innerhalb eines Zeitfensters induziert eine Spannung, die proportional zur Lichtmenge, d. h. zur Intensität ist. Durch das Quantisieren dieser Spannung gelingt der Übergang zu einem diskreten Wert, der von Computern verarbeitet werden kann. Die Anordnung mehrerer Einzelsensoren zu einer Matrix ergibt eine diskrete zweidimensionale Funktion $\mathbf{I}(u, v)$, die als digitales monochromatisches Bild interpretierbar ist. Durch die Verwendung einer Bayer-Maske sind Farbinformationen extrahierbar, wodurch sich ein digitales RGB-Bild ergibt.

2.1.1 Bilder als zweidimensionale Signale

Die diskrete Darstellung ermöglicht die maschinelle Verarbeitung mit klassischer zweidimensionaler Signalverarbeitung. Eines der relevantesten Werkzeuge der digitalen Bildverarbeitung ist die diskrete Faltung eines Bilds $\mathbf{I}(u,v)$ mit einem diskreten, linearen und ortsinvarianten System $\mathbf{k}(u,v)$, dem sog. Filter. Die Faltung ist definiert als

$$(\mathbf{I} ** \mathbf{k})(u,v) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \mathbf{I}(u-k, v-l) \cdot \mathbf{k}(k,l). \tag{2.1}$$

Durch die gezielte Auswahl eines geeigneten Filters können bestimmte Bildeigenschaften verstärkt oder vermindert werden. So können durch ein Tiefpassfilter wie bspw. einem Gauß- oder Rechteckfilter hohe Ortsfrequenzen gedämpft werden, was zu verschwommenen Details und Kanten führt, aber auch mittelwertfreies Bildrauschen vermindert. Hochpassfilter dagegen verstärken Kanten und Rauschen gleichermaßen. Der Marr-Hildreth-Operator [112] kombiniert einen approximativen Differenzierungsoperator zweiten Grades mit einem Tiefpassfilter, um Bildkanten zu exponieren und gleichzeitig Rauschen zu unterdrücken.

Mit der Bildverschärfung wird der visuelle Schärfeeindruck eines Bildes dadurch verstärkt, dass tieffrequente Anteile vom Originalbild subtrahiert werden. Im Detail wird hier das mit α_{BS} gewichtete und mit einem Tiefpass gefilterte Bild vom Originalbild subtrahiert und anschließend durch $1-\alpha_{\mathrm{BS}}$ geteilt.

Neben der Filterung stellt die Abtastratenkonvertierung eine relevante Operation der digitalen Signalverarbeitung dar. Hierbei wird die Anzahl der Bildelemente (engl. *pixel*) erhöht (Interpolation, engl. *upsampling*) bzw. verringert (Dezimierung, engl. *downsampling*). Um Störeinflüsse bei der Signalrekonstruktion (Interpolation) bzw. durch spektrale Überfaltung (engl. *aliasing*, Dezimierung) zu reduzieren, ist eine Faltung des Signals mit einem Tiefpassfilter, dem sog. Interpolationsfilter $\mathbf{k}_{\mathrm{Int}}$ bzw. *Antialiasing*-Filter $\mathbf{k}_{\mathrm{AA}}$, notwendig. Die Dezimierung $(\cdot)_{\downarrow M}$ eines Bildes $\mathbf{I}$ um den Faktor $M \in \mathbb{N}$ ist definiert durch

$$(\mathbf{I}(u,v))_{\downarrow M} = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \mathbf{I}(uM-k, vM-l) \cdot \mathbf{k}_{\mathrm{AA}}(k,l). \tag{2.2}$$

Während eine Dezimierung zur Erhöhung der Informationsdichte verwendet wird, dient eine Interpolation meist der Anpassung des Signals

an ein System, das auf einer höheren Auflösung operiert. Die Interpolation $(\cdot)_{\uparrow M}$ um den Faktor $M \in \mathbb{N}$ besteht aus zwei Schritten. Zuerst wird das Signal bzw. Bild expandiert, indem zwischen jedem Abtastwert $M-1$ Nullen eingefügt werden. Das resultierende Signal wird anschließend mit einem Interpolationsfilter mit Tiefpasscharakteristik gefaltet.

2.1.2 Nichtlineare digitale Bildverarbeitung

Abseits der Bildverarbeitung mittels linearer Systeme werden häufig nichtlineare Operationen zur Bildverbesserung verwendet, welche sich für die Vorverarbeitung und Augmentierung (s. Kapitel 3) der Eingangsbilder von konvolutionären neuronalen Netzen eignen. Im Folgenden werden die wichtigsten Operationen kurz erläutert, welche, sofern nicht anders gekennzeichnet, aus [11] entnommen sind.

Bei der Histogrammspreizung wird der Kontrast eines Bildes dadurch erhöht, dass das geschätzte Histogramm der Grauwerte linear gestreckt wird, um den gesamten verfügbaren Wertebereich zu nutzen. Die Histogrammegalisierung dagegen bildet das geschätzte Histogramm auf eine Gleichverteilung ab. CLAHE (*contrast limited adaptive histogram equalization*) nach [65] egalisiert das Histogramm kleiner Bildausschnitte, die typischerweise eine Seitenlänge von 4 bis 64 Pixeln, der sog. Kachelgittergröße, haben. Dies erhöht den lokalen Kontrast.

Homomorphe Filter vermindern den Effekt infolge inhomogener Beleuchtung, die multiplikativ mit dem Nutzsignal des Eingangsbildes verknüpft ist, indem ein Hochpassfilter auf das logarithmische Eingangsbild angewendet wird. Anschließend wird das resultierende Bildsignal potenziert. Bei Bilateralfiltern wird das Bild unter der Aufrechterhaltung der Kanteninformationen geglättet. Bei der Homogenisierung ersten Grades wird der lokale Mittelwert eines Bildes eliminiert, sodass der Mittelwert unabhängig von der Bildposition ist. Allgemein werden beim Grad m die ersten m lokalen stochastischen Momente des Bildes entfernt. Die lokale Mittelwertbildung wird durch eine Tiefpassfilterung realisiert.

Medianfilter werden insbesondere zum Unterdrücken von Impulsrauschen verwendet, da das Medianfilter einem Bildpunkt den Medianwert seiner Umgebung zuordnet.

Bei affinen Transformationen werden die Pixelkoordinaten eines Bildes durch eine Matrix und einen Translationsvektor auf eine neue Position

abgebildet. Durch eine anschließende Interpolation sind Drehungen, Scherungen, Translationen und Vergrößerungen bzw. Verkleinerungen von Bildern möglich.

Durch die Darstellung des Bildes im HSV-Farbraum können Helligkeit, Sättigung und der Farbwert einfach manipuliert werden.

2.2 Grundlagen künstlicher neuronaler Netze

Im Folgenden werden die Grundlagen von neuronalen Netzen beschrieben. Ausgehend vom Neuron als kleinstes Element wird der Aufbau und das Training von neuronalen Netzen erläutert.

2.2.1 Einlagiges Perzeptron

Das künstliche Neuron bildet den Ausgangspunkt für das einlagige Perzeptron und für neuronale Netze allgemein. Der Ursprung des künstlichen Neurons liegt in der Beobachtung aus dem Bereich der Neurowissenschaften, dass sich aus biologischen Neuronen durch ihr Alles-oder-nichts-Prinzip eine Aussagenlogik ableiten lässt [115]. Ein künstliches Neuron ist in Abb. 2.1 schematisch dargestellt. Aus den Eingängen $\mathbf{x}$ und den Gewichten $\mathbf{h}_i$ des Neurons i wird zunächst das Innenprodukt gebildet. Um einen Bias-Term b_i zu integrieren, wird ein zusätzlicher Eingang mit dem Wert 1 verwendet. Die Ausgabe des Neurons ergibt sich aus der Aktivierung des Innenproduktes mit der nichtlinearen Aktivierungsfunktion $\sigma_{\text{akt}}(\cdot)$ mit $y_i = \sigma_{\text{akt}}(\langle \mathbf{x}, \mathbf{h}_i \rangle + b_i)$, die das Alles-oder-nichts-Prinzip des biologischen Neurons nachahmt.

Durch das Parallelschalten mehrerer Neuronen mit den gleichen Eingangswerten, aber verschiedenen Gewichten ergibt sich das einlagige Perzeptron nach [129], welches als Schicht innerhalb eines neuronalen Netzes einsetzbar ist. Der Ausgabevektor $\mathbf{y}$ ist als Matrix-Vektor-Produkt zwischen den Gewichten $\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_N]^\mathrm{T}$ und Bias-Termen $\mathbf{b}$ aller N Neuronen und dem Eingangsvektor $\mathbf{x}$ mit anschließender Aktivierung darstellbar durch

$$\mathbf{y} = \sigma_{\text{akt}}(\mathbf{H}\mathbf{x} + \mathbf{b}). \tag{2.3}$$

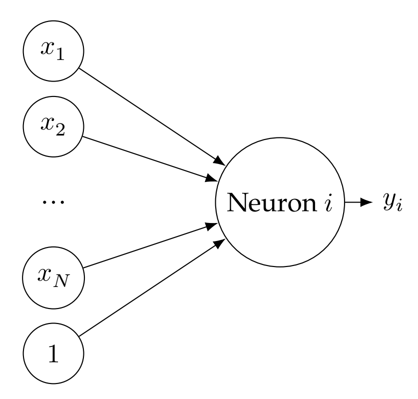


Abbildung 2.1 Schematischer Aufbau des Neurons. Die Eingangsdaten $\mathbf{x}$ werden im Neuron mit den Gewichten multipliziert, wobei der Wert 1 mit dem Biasterm b_i multipliziert wird. Die Produkte werden anschließend aufsummiert und aktiviert, sodass sich die Ausgabe y_i ergibt.

2.2.2 Aufbau von neuronalen Netzen

Das mehrlagige Perzeptron ist der Ausgangspunkt für den Aufbau neuronaler Netze und wird durch die Verkettung mehrerer einlagiger Perzeptronen gebildet, die im Folgenden als Schichten bezeichnet werden. Abb. 2.2 zeigt ein Beispiel eines zweilagigen Perzeptrons mit drei Eingängen, fünf verborgenen Neuronen und zwei Ausgangsneuronen. Während beim klassischen Perzeptron vollständig besetzte Gewichtsmatrizen $\mathbf{H}$ und ein binäres Schwellenwertverfahren als Aktivierung verwendet werden, werden in modernen neuronalen Netzen durch gezielte Restriktionen der Gewichtsmatrizen und der Wahl anderer Aktivierungen bestimmte Operationen der digitalen Signalverarbeitung wie Faltungen oder Abtastratenkonvertierung nachgeahmt [92]. Im Folgenden werden die wichtigsten Schichten moderner CNNs kurz aufgeführt.

Durch die Interpretation von $\mathbf{x}$ in Gleichung 2.3 als ein vektorisiertes Bild sind zweidimensionale Faltungen wie in Abschnitt 2.1 durch $\mathbf{H}$ darstellbar. Hierbei ist $\mathbf{H}$ eine dünnbesetzte Faltungsmatrix. Sowohl das Eingangsbild $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_D] \in \mathbb{R}^{(H,B,D)}$ als auch die Aktivierungen $\mathbf{Y} = [\mathbf{Y}_1, \ldots, \mathbf{Y}_L] \in \mathbb{R}^{(H,B,L)}$ einer Schicht werden hierfür als dreidimensionale Tensoren dargestellt. Digitale Bilder $\mathbf{I}(u, v)$ sind i. A. in die

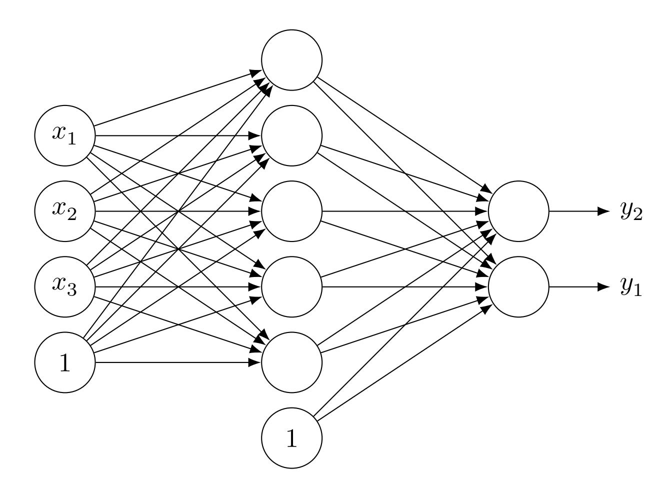


Abbildung 2.2 Schematischer Aufbau des zweischichtigen Perzeptrons.

Darstellung als Eingangstensor überführbar, wobei $D = 1$ für monochromatische und $D = 3$ für Farbbilder gilt.

In zweidimensionalen Faltungsschichten (engl. *convolutional layers*) wird jede der D Aktivierungskarten von $\mathbf{X}$ mit insgesamt L verschiedenen Filterkernen $\mathbf{k}_l = [\mathbf{k}_{l,1}, \dots, \mathbf{k}_{l,D}] \in \mathbb{R}^{(3,3,L,D)}$, $l = 1, \dots, L$ der Größe 3×3, welche sich aus Gründen der Recheneffizienz und Performance durchgesetzt hat, und Tiefe D gefaltet und mit einem Bias-Term b_l, $l = 1, ..., L$ addiert. Anschließend wird der resultierende Tensor entlang der Tiefe aufsummiert. Die Werte der Filterkerne werden während der Lernphase bestimmt. Die Rechenvorschrift für die Faltung einer einzelnen Aktivierungskarte der Ausgabe $\mathbf{Y}$ lautet

$$\mathbf{Y}_l = \sigma_{\text{akt}} \left(\sum_{d=1}^{D} \mathbf{X}_d ** \mathbf{k}_{l,d} + b_l \right) . \tag{2.4}$$

Durch Anhängen von Nullen zum Aufrechterhalten der örtlichen Dimension (engl. *zero-padding*) wird ein Eingangstensor der Dimension $H \times B \times D$ auf einen Ausgangstensor der Größe $H \times B \times L$ abgebildet. Der Vorgang ist in Abb. 2.3 für $D = 3$ und $L = 4$ illustriert und benötigt insgesamt $10 \cdot HBLD$ MAC-Operationen (engl. *multiply-accumulate operations*) für die Ausführung.

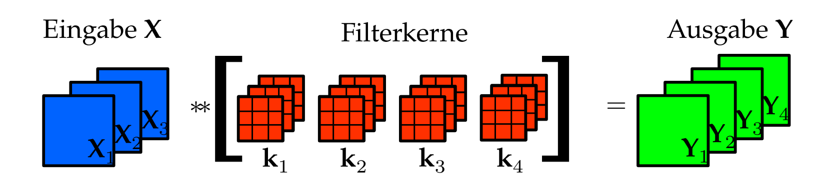


Abbildung 2.3 Illustration der Faltung eines CNN. Auf die Darstellung der Addition der Biasterme wurde in der Abbildung der Übersichtlichkeit wegen verzichtet.

Für die Abtastratenkonvertierung werden Hochtastschichten (engl. *upsampling layers*) und *Pooling*-Schichten verwendet. Diese vergrößern bzw. verkleinern die ersten beiden Dimensionen des Eingangstensors. Die *Max-Pooling*-Schicht mit dem Dezimierungsfaktor 2 und der Schrittweite 2 wird aufgrund ihres Alles-oder-nichts-Verhaltens üblicherweise zum Dezimieren eingesetzt [136]. Da die Ausgabe dieser Schicht jeweils das Maximum von (2×2)-Elementen ist und kein *Antialiasing*-Filter verwendet wird, ist die Operation dieser Schicht nicht invertierbar und nichtlinear, wodurch die aggregierten Merkmale im Allgemeinen translationsvariant sind [4]. *Max-Pooling* um den Faktor M wird im Folgenden mit $(\bullet)_{\downarrow M,\mathrm{MP}}$ bezeichnet. Die Operation der Hochtastschichten folgt der Beschreibung gemäß Abschnitt 2.1. In der Lernphase wird für jede Karte $\mathbf{X}_d$ der Eingabe ein eigenes Interpolationsfilter $\mathbf{k}_{\mathrm{Int},d}$ ermittelt.

Als Aktivierung der vorderen und intermediären Faltungsschichten wird im Folgenden die *leaky rectifier linear unit* (kurz: leaky ReLU) $\sigma_{\mathrm{akt}}(x) = \max\{a \cdot x, b \cdot x\}$ mit $a = 0$ und $b = 1$ verwendet. Diese Funktion wird auch als ReLU bezeichnet. Für Multi-Klassen-Klassifikationsaufgaben wird als Aktivierung der letzten Schicht eine *softmax*-Funktion und für Segmentierungsaufgaben sowie Zwei-Klassen-Klassifikationsaufgaben eine *sigmoid*-Funktion verwendet.

Des Weiteren wird eine Chargennormalisierungschicht (engl. *batch normalization layer*, kurz: BN) nach [77] verwendet, um den Effekt instabiler Gradienten in der Lernphase abzumildern. Hierbei wird jede Charge $\mathcal{B}$ (engl. *(mini-) batch*) und jede Aktivierungskarte l durch den geschätzten Mittelwert $\hat{\mu}_{\mathrm{BN},\mathcal{B},l}$ und die geschätzte Standardabweichung $\hat{\sigma}_{\mathrm{BN},\mathcal{B},l}$ standardisiert und anschließend mit $\gamma_{\mathrm{BN},l}$ und $\delta_{\mathrm{BN},l}$ skaliert und verschoben.

Die Rechenvorschrift für eine Aktivierungskarte einer Faltungsschicht mit ReLU-Aktivierung, BN und einer *Max-Pooling*-Schicht lautet

$$\mathbf{Y}_l = \sigma_{\text{akt}}\left(\gamma_{\text{BN},l} \cdot \frac{\sum_{d=1}^{D} \mathbf{X}_d ** \mathbf{k}_{l,d} + b_l - \hat{\mu}_{\text{BN},\mathcal{B},l}}{\hat{\sigma}_{\text{BN},\mathcal{B},l} + \epsilon} + \delta_{\text{BN},l}\right)_{\downarrow 2,\text{MP}} . \tag{2.5}$$

Im Folgenden wird die Menge aller trainierbaren Gewichte eines neuronalen Netzes als $\mathcal{H}$ bezeichnet. Die Abbildung des neuronalen Netzes wird mit $\mathrm{f} : \mathcal{X} \rightarrow \hat{\mathcal{P}}$ bezeichnet und bildet den Eingabedatensatz $\mathcal{X}$ auf die Ausgabe $\hat{\mathcal{P}}$ ab. Die Ausgabe $\hat{\mathcal{P}}$ stellt somit eine Schätzung der Grundwahrheit $\mathcal{M}$ des Datensatzes dar.

2.2.3 Training von neuronalen Netzen

Während des Trainings werden die trainierbaren Gewichte $\mathcal{H}$ des neuronalen Netzes mit Hilfe des Lerndatensatzes $\{\mathcal{X}_\mathrm{L}, \mathcal{M}_\mathrm{L}\}$ erlernt. Für Bilddatensätze ist $\mathcal{X}$ eine Menge an Bildern und $\mathcal{M}$ die dazugehörige Grundwahrheit, welche im Falle der Klassifikation die dominante Klasse des Bildes indiziert. Die Gewichte werden schrittweise mit einem geeigneten Gradientenabstiegsverfahren berechnet, um die Zielfunktion $\mathcal{L}(\mathrm{f}(\mathcal{X}_\mathrm{L}; \mathcal{H}), \mathcal{M}_\mathrm{L})$ zu minimieren. Falls nicht anders angegeben, wird im Folgenden der NADAM-Optimierer als Gradientenabstiegsverfahren benutzt, der den ADAM-Optimierer [85] um ein Nesterov-Moment erweitert. Für Klassifikationsaufgaben wird im Folgenden die kategorische bzw. binäre Kreuzentropie als Zielfunktion verwendet.

Das Training besteht aus Durchläufen (engl. *epochs*). Bei einem Durchlauf wird jedes Bild des Lerndatensatzes $\mathcal{X}_\mathrm{L}$ im Mittel einmal in das CNN eingegeben. Dabei wird eine Menge von Bildern zu einer Charge $\mathcal{B}$ zusammengefasst. Für jede Charge wird ein Gradientenschritt berechnet, um anschließend damit die Gewichte $\mathcal{H}$ zu aktualisieren. Eine Aktualisierung mittels einer Charge wird im Folgenden als Iteration bezeichnet. Mit der Chargengröße $N_\mathcal{B} = |\mathcal{B}|$ ergeben sich $\lfloor |\mathcal{X}_L| / N_\mathcal{B} \rfloor + 1$ Iterationen pro Durchlauf.

Eine größere Chargengröße führt somit zu einer kleineren Anzahl an Iterationen, was in einem rechenineffizienten Training und in einer schlechteren Fähigkeit zum Generalisieren auf ungesehene Daten der

neuronalen Netze resultiert [50, 68, 84, 90, 105, 113, 117]. Als Maß für die Generalisierbarkeit wird die Generalisierungslücke Γ (engl. *generalization gap*) verwendet, die sich aus der Differenz zwischen den erzielten Genauigkeiten von Lern- und Testdatensatz ergibt. Um Γ möglichst klein zu halten, ist es erforderlich, das CNN zu regularisieren oder dem neuronalen Netz in der Trainingsphase Invarianzen bzgl. bestimmter Variationen beizubringen, da CNNs sehr sensibel gegenüber kleinsten Variationen in den Bilddaten wie bspw. Translation [4], Skalierung [121] oder Kontrast [67] sein können. Geschieht dies nicht, besteht die Gefahr der Überanpassung, d. h. das CNN lernt die Manifestationen bestimmter Merkmale im Lerndatensatz auswendig und kann diese nicht auf neue Daten übertragen.

Um dies zu vermeiden, müssen entweder Invarianzen explizit durch Augmentieren und Vorverarbeitung beigebracht werden oder es müssen Regularisierungsverfahren nach [51, 92, 149] wie L_1- bzw. L_2-Regularisierung der Gewichte, vorzeitiges Anhalten (engl. *early stopping*), Kombination mehrerer CNNs, Dropout oder Einfügen von Rauschen verwendet werden. In dieser Arbeit wird neben Vorverarbeitung (s. Abschnitt 3.2) und Augmentierung (s. Abschnitt 3.3) zusätzlich eine L_1- bzw. L_2-Regularisierung verwendet. In Abschnitt 3.4 wird darüber hinaus Rauschen als Form der Regularisierung verwendet.

Bei der L_2-Regularisierung wird die Zielfunktion $\mathcal{L}$ um einen Term $\mathcal{L}_2$ erweitert, der proportional zur L_2-Norm aller Gewichte ist. Es gilt $\mathcal{L}_2 = \lambda_2 \cdot \sum_{h \in \mathcal{H}} |h|^2$. Entsprechend wird für eine L_1-Regularisierung die gewichtete Summe der Absolutbeträge der Gewichte als Strafterm hinzugefügt.

Bei der Augmentierung wird der Datensatz künstlich durch eine stochastische Transformation der Bilder erweitert [141]. Zwar kann Augmentierung auch in der Merkmals- oder Annotierungsdomäne durchgeführt werden [13, 19, 76, 98, 156, 169, 176], jedoch beschränkt sich diese Arbeit auf die Augmentierung in der Bilddomäne. Hierbei bleibt die Annotierung unverändert und die Bilder werden durch eine Kaskade von Bildtransformationen verändert. Zu den Transformationen gehören u. a. Spiegelung [92], Verzerrungen [35], Farbtransformationen [30] oder andere affine Transformationen.

Im Gegensatz dazu werden Vorverarbeitungs- bzw. Bildverbesserungsverfahren deterministisch eingesetzt, um bestimmte Variationen von Merkmalen in den Eingabebildern zu unterdrücken.

2.3 Grundlagen der Robotik und der bildbasierten Regelung

In diesem Abschnitt wird ein Robotersystem für eine bildbasierte Regelung vorgestellt. Dazu wird in Abschnitt 2.3.1 sowohl die Tiefenbildkamera als auch der bewegbare Roboterarm, an dessen Endeffektor die Kamera montiert ist, modellartig beschrieben. Dies beinhaltet auch die Kalibrierung und Transformation zwischen den Koordinatensystemen beider Hardwarekomponenten.

Im darauffolgenden Abschnitt 2.3.2 wird die klassische bildbasierte Regelung mit Punktmerkmalen und ihren Erweiterungen kurz thematisiert.

2.3.1 Aufbau von Robotersystemen

In Abb. 2.4 ist der schematische Aufbau eines geeigneten Roboterarms mit einer Kamera am Endeffektor dargestellt. Der Roboterarm verfügt über eine feste Anzahl N_q an Gelenken $q_i,\ i = 0, ..., N_\mathrm{q} - 1$, die jeweils genau zwei starre Glieder miteinander verbinden. Im Folgenden werden nur ideale Drehgelenke betrachtet, wodurch q_i mit einem Winkel korrespondiert. Bei Kenntnis über die Maße der Glieder kann die Position $\mathbf{x}_\mathrm{E}^\mathrm{W} = [x_\mathrm{E}^\mathrm{W}, y_\mathrm{E}^\mathrm{W}, z_\mathrm{E}^\mathrm{W}]$ und Orientierung $\boldsymbol{\phi}_\mathrm{E}^\mathrm{W} = [\phi_\mathrm{E}^\mathrm{W}, \theta_\mathrm{E}^\mathrm{W}, \psi_\mathrm{E}^\mathrm{W}]$ des Endeffektors, welche im Folgenden unter dem Begriff Lage zusammengefasst werden, abhängig von den aktuellen Gelenkwinkeln $\mathbf{q} = [q_0, ..., q_{N_\mathrm{q}-1}]$ berechnet werden [12]. Gemäß [111] wird die Berechnungsvorschrift für die Lage des Endeffektors $(\mathbf{x}_\mathrm{E}^\mathrm{W}, \boldsymbol{\phi}_\mathrm{E}^\mathrm{W})$ aus den Gelenkwinkeln $\mathbf{q}$ als *direkte Kinematik* bezeichnet. Die Gesamttransfomation $\tilde{\mathbf{T}}_{0,N_\mathrm{q}}(\mathbf{q})$ ergibt sich aus der Verkettung der Einzeltransformationen $\tilde{\mathbf{T}}_{i,i+1}(q_i)$ des i-ten Gelenks. $\tilde{\mathbf{T}}_{0,N_\mathrm{q}}(\mathbf{q})$ ist eine (4×4)-Matrix, die die lineare Koordinatentransformation zwischen der Basis des Roboters und dem Endeffektor in homogenen Koordinaten beschreibt. Die Matrix enthält eine Verkettung

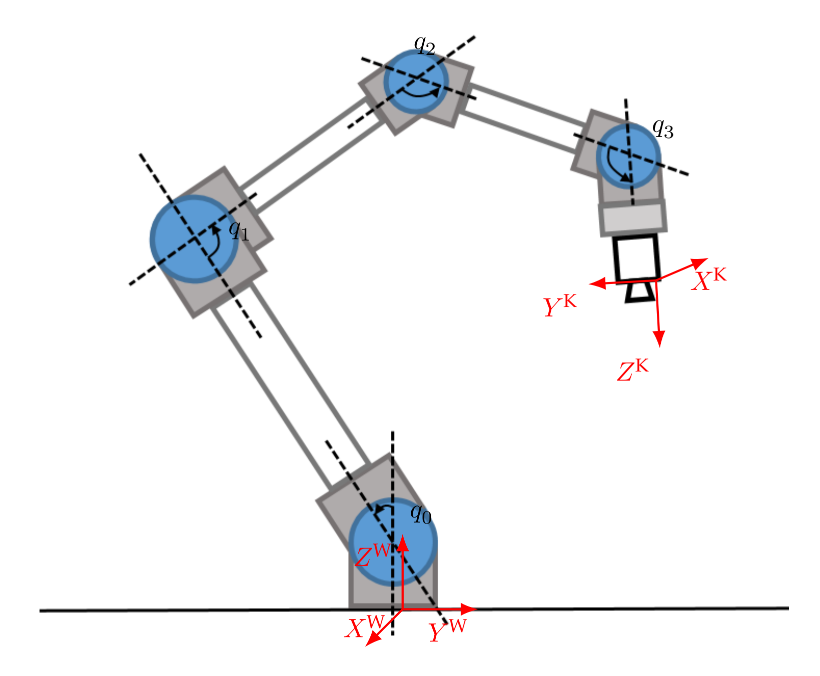


Abbildung 2.4 Schematischer Aufbau des Robotersystems mit $N_q = 4$ Gelenken.

der drei Rotationsmatrizen $\mathbf{R}(\phi_E^W) = \mathbf{R}_z(\psi_E)\mathbf{R}_y(\theta_E)\mathbf{R}_x(\phi_E)$ und dem Translationsvektor $\mathbf{x}_e^W$. Die Transformationsmatrix hat die allgemeine Form

$$\tilde{\mathbf{T}}_{0,N_q}(\mathbf{q}) = \left[\begin{array}{c|c} \mathbf{R}(\phi_E^W) & \mathbf{x}_E^W \\ \hline \mathbf{0} & 1 \end{array}\right]. \tag{2.6}$$

Wird der Ursprung des Weltkoordinatensystems in die Basis des Roboters gelegt, beschreibt $\tilde{\mathbf{T}}_{0,N_q}(\mathbf{q})$ ferner die Lage des Endeffektors.

Da die Abbildung $q_i \mapsto \tilde{\mathbf{T}}_{i,i+1}(q_i)$ auf Grundlage der Geometrie des Roboters berechnet wird [111], ist die Abbildung im Allgemeinen nichtlinear und injektiv. Daher ist die Lage des Endeffektors bei Kenntnis von $\mathbf{q}$ eindeutig, während die Berechnung geeigneter Gelenkwinkel bei einer gewünschten Lage, der sog. *inversen Kinematik*, eine nicht triviale Aufgabe ist, da u. a. Kollisionen zwischen Roboterkomponenten und Singularitäten zu vermeiden sind. Diese Thematik wird in der Forschung

ausführlich diskutiert [3, 49, 110, 111]. Moderne Roboterarme werden üblicherweise vom Hersteller bzgl. $\tilde{\mathbf{T}}_{0,N_\mathrm{q}}(\mathbf{q})$ kalibriert und verfügen über eine interne Berechnung der inversen Kinematik.

Durch eine dynamische Betrachtung der Kinematik sind Bewegungen zwischen einer Startlage $(\mathbf{x}^\mathrm{W}_{\mathrm{E},1}, \phi^\mathrm{W}_{\mathrm{E},1})$ und einer Endlage $(\mathbf{x}^\mathrm{W}_{\mathrm{E},2}, \phi^\mathrm{W}_{\mathrm{E},2})$ ableitbar [87]. Diese ist entweder durch die Differenz der Lageparameter $(\Delta\mathbf{x}^\mathrm{W}, \Delta\phi^\mathrm{W})$ oder durch eine Transformationsmatrix $\mathbf{T}_{1,2}$ nach dem Schema aus Gleichung 2.6 beschreibbar. Die Lage nach einer Bewegung ist über $\tilde{\mathbf{T}}_{0,N_\mathrm{q}}(\mathbf{q})\mathbf{T}_{1,2}$ berechenbar. Im Folgenden werden ausschließlich lineare Bewegungen des Roboterarms betrachtet.

Für die Versuche in dieser Arbeit ist eine Tiefenkamera am Endeffektor montiert. Insbesondere bei der bildbasierten Regelung werden Transformationen bezüglich der Kamerakoordinaten $\mathbf{X}^\mathrm{K} = (X^\mathrm{K}, Y^\mathrm{K}, Z^\mathrm{K})$ angegeben (s. Abb. 2.4). Durch eine sog. *eye-in-hand*-Kalibrierung wird die Lagetransformation $\tilde{\mathbf{T}}_{\mathrm{E,K}}$ zwischen Endeffektor und Kamera bestimmt. Gemäß [164] wird hierfür ein Schachbrettmuster aus mehreren Kameralagen aufgenommen. Die Transformationsmatrix $\tilde{\mathbf{T}}_{\mathrm{W,K}}(\mathbf{q}) = \tilde{\mathbf{T}}_{0,N_\mathrm{q}}(\mathbf{q})\tilde{\mathbf{T}}_{\mathrm{E,K}}$ beschreibt nicht nur die Lage der Kamera, sondern ist auch für die Umrechnung von Welt- in Kamerakoordinaten nutzbar.

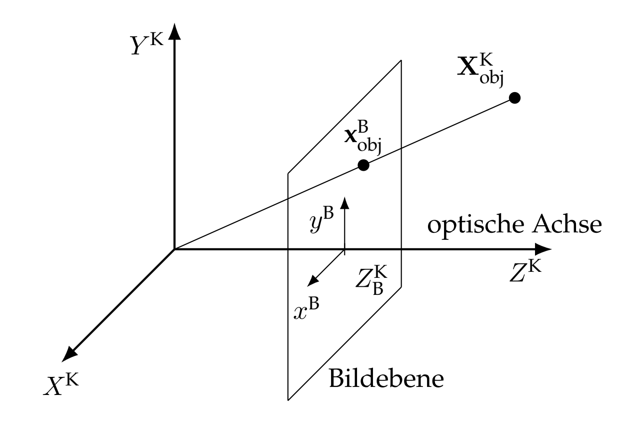


Abbildung 2.5 Verwendetes Lochkameramodell.

Zur Beschreibung der Kamera wird das Lochkameramodell aus Abb. 2.5 verwendet. Die Lichtstrahlen des Objekts an der Position $\mathbf{X}_{\mathrm{obj}}^{\mathrm{K}}$ durchdringen die Bildebene, die sich im Abstand $Z_{\mathrm{B}}^{\mathrm{K}}$, der sog. Kammerkonstante, zum Ursprung der Kamera befindet, im Punkt $\mathbf{x}_{\mathrm{obj}}^{\mathrm{B}}$. Mithilfe des Strahlensatzes ergibt sich

$$\begin{bmatrix} x_{\mathrm{obj}}^{\mathrm{B}} \\ y_{\mathrm{obj}}^{\mathrm{B}} \end{bmatrix} = \frac{Z_{\mathrm{B}}^{\mathrm{K}}}{Z_{\mathrm{obj}}^{\mathrm{K}}} \begin{bmatrix} X_{\mathrm{obj}}^{\mathrm{K}} \\ Y_{\mathrm{obj}}^{\mathrm{K}} \end{bmatrix} . \tag{2.7}$$

Mit der Transformation in homogene Koordinaten $[{}^{a}/_{c}, {}^{b}/_{c}]^{\mathrm{T}} \mapsto [a, b, c]^{\mathrm{T}}$ wird der Strahlensatz zu

$$\begin{bmatrix} x_{\mathrm{obj}}^{\mathrm{B}} \\ y_{\mathrm{obj}}^{\mathrm{B}} \\ Z_{\mathrm{B}}^{\mathrm{K}} \end{bmatrix} = \begin{bmatrix} X_{\mathrm{obj}}^{\mathrm{K}} \\ Y_{\mathrm{obj}}^{\mathrm{K}} \\ Z_{\mathrm{obj}}^{\mathrm{K}} \end{bmatrix} . \tag{2.8}$$

Durch die Hauptpunktkorrektur und Quantisierung ergibt sich der fundamentale Zusammenhang zwischen den Sensorkoordinaten des Bildes $\mathbf{u} = [u, v]^{\mathrm{T}}$ und der Position $\mathbf{X}_{\mathrm{obj}}^{\mathrm{K}}$ eines Objektes im Raum zu

$$\begin{aligned} \begin{bmatrix} u \\ v \\ Z_{\mathrm{B}}^{\mathrm{K}} \end{bmatrix} &= \begin{bmatrix} \Delta_{\mathrm{x}}^{-1} & 0 & \frac{u_0}{Z_{\mathrm{B}}^{\mathrm{K}}} \\ 0 & \Delta_{\mathrm{y}}^{-1} & \frac{v_0}{Z_{\mathrm{B}}^{\mathrm{K}}} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{\mathrm{obj}}^{\mathrm{K}} \\ Y_{\mathrm{obj}}^{\mathrm{K}} \\ Z_{\mathrm{obj}}^{\mathrm{K}} \end{bmatrix} \\ \Rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \begin{bmatrix} f_{\mathrm{x}} & 0 & u_0 \\ 0 & f_{\mathrm{y}} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{\mathrm{obj}}^{\mathrm{K}} \\ Y_{\mathrm{obj}}^{\mathrm{K}} \\ Z_{\mathrm{obj}}^{\mathrm{K}} \end{bmatrix} =: \mathbf{C}_{\mathrm{K}} \mathbf{X}_{\mathrm{obj}}^{\mathrm{K}} . \end{aligned} \tag{2.9}$$

Die Matrix $\mathbf{C}_{\mathrm{K}}$ mit $Z_{\mathrm{B}}^{\mathrm{K}} \Delta_x^{-1} = f_x$ und $Z_{\mathrm{B}}^{\mathrm{K}} \Delta_y^{-1} = f_y$ wird als Kameramatrix bezeichnet. Für eine feste Fokussierung der Kamera ist $\mathbf{C}_{\mathrm{K}}$ konstant und durch eine Kalibrierung mit einem Schachbrettmuster aus verschiedenen Perspektiven bestimmbar [177]. Bei Tiefenbildkameras, die auf Streifenlichtprojektionen beruhen, wie bspw. `Zivid`-Kameras ist eine Selbstkalibrierung während der Aufnahme möglich [174].

2.3.2 Klassische bildbasierte Regelung

Das Ziel der bildbasierten Regelung besteht darin, eine bewegliche Kamera in eine Position zu bringen, in der der Fehler $\mathbf{e}_{\mathrm{m}}(t) = \mathbf{m}(t) - \mathbf{m}^*$

zwischen einem Bildmerkmalsvektor $\mathbf{m}(t)$, der aus dem Bild der Kamera zum Zeitpunkt t ableitbar ist, und einem Zielmerkmalsvektor $\mathbf{m}^*$ betragsmäßig minimal ist. Die bewegliche Kamera wird dabei über einen Roboter in Echtzeit durch eine Rückkopplungsschleife gesteuert [10, 17, 75, 168]. Der in dieser Arbeit verwendete Roboteraufbau ist in Abb. 2.4 schematisch dargestellt.

Nach [75] können Bildmerkmale beliebige Strukturmerkmale sein, die aus dem Bild extrahiert werden können. Die reellwertigen Größen, die aus dem Bildmerkmalen ableitbar sind, werden gemäß [81] als Merkmalsparameter m bezeichnet und bilden den Merkmalsvektor $\mathbf{m}$. Besonders geeignet sind Merkmale, die aus verschiedenen Lagen der Kamera sichtbar sind wie bspw. Punktkoordinaten [42], Abstände und Orientierungen zwischen Punkten [43], Kantenlängen und Flächen [167] oder Parameter anderer geometrischer Figuren [42, 167].

Für die bildbasierte Regelung ist der Zusammenhang zwischen der translatorischen Bewegung $\mathbf{v}^{\mathrm{K}} = [v_x^{\mathrm{K}}, v_y^{\mathrm{K}}, v_z^{\mathrm{K}}]^{\mathrm{T}}$ bzw. den Winkelgeschwindigkeiten $\boldsymbol{\omega}^{\mathrm{K}} = [\omega_x^{\mathrm{K}}, \omega_y^{\mathrm{K}}, \omega_z^{\mathrm{K}}]^{\mathrm{T}}$ der Kamera und der korrespondierenden Änderung der Bildeigenschaften $\mathbf{m}$ relevant. Dieser Zusammenhang ist über die Bild-Jacobi-Matrix $\mathbf{J}$ nach [168] mit

$$\dot{\mathbf{m}} = \mathbf{J} \begin{bmatrix} \mathbf{v}^{\mathrm{K}} \\ \boldsymbol{\omega}^{\mathrm{K}} \end{bmatrix} \tag{2.10}$$

gegeben. Die Jacobi-Matrix ist bestimmbar durch die optische Abbildung aus Gleichung 2.9 und den Bewegungsgleichungen für einen Punkt $\mathbf{X}^{\mathrm{K}}$ in Kamerakoordinaten [75]

$$\begin{aligned} \dot{X}^{\mathrm{K}} &= Z^{\mathrm{K}}\omega_y^{\mathrm{K}} - Y^{\mathrm{K}}\omega_z^{\mathrm{K}} + v_x^{\mathrm{K}} \\ \dot{Y}^{\mathrm{K}} &= X^{\mathrm{K}}\omega_z^{\mathrm{K}} - Z^{\mathrm{K}}\omega_x^{\mathrm{K}} + v_y^{\mathrm{K}} \\ \dot{Z}^{\mathrm{K}} &= Y^{\mathrm{K}}\omega_x^{\mathrm{K}} - X^{\mathrm{K}}\omega_y^{\mathrm{K}} + v_z^{\mathrm{K}} \end{aligned} \tag{2.11}$$

bzw.

$$\dot{\mathbf{X}}^{\mathrm{K}} = \boldsymbol{\omega}^{\mathrm{K}} \times \mathbf{X}^{\mathrm{K}} + \mathbf{v}^{\mathrm{K}}, \tag{2.12}$$

wobei mit $\times$ das Kreuzprodukt bezeichnet wird. Die Jacobi-Matrix wird üblicherweise für Koordinaten $\mathbf{x}_1^{\mathrm{B}} = [x_1^{\mathrm{B}}, y_1^{\mathrm{B}}, 1]^{\mathrm{T}}$ auf einer normalisierten

imaginären Bildebene im Abstand $Z_{\mathrm{B}}^{\mathrm{K}} = 1$ zum optischen Zentrum angegeben, um unabhängig von Kameraparametern zu sein. Die Umrechnung in Sensorkoordinaten $[u, v, 1]^{\mathrm{T}}$ gelingt mithilfe von Gl. (2.8) und (2.9) bei Kenntnis von $\mathbf{C}_{\mathrm{K}}$.

Der Zusammenhang zwischen einem Punktmerkmal $\mathbf{x}_1^{\mathrm{B}}$ und den Bewegungen der Kamera ergibt sich mithilfe der Differenzierung von Gl. (2.7) zu

$$\dot{x}_1^{\mathrm{B}} = \frac{\dot{X}^{\mathrm{K}} Z^{\mathrm{K}} - \dot{Z}^{\mathrm{K}} X^{\mathrm{K}}}{Z_{\mathrm{K}}^2} \text{ und } \dot{y}_1^{\mathrm{B}} = \frac{\dot{Y}^{\mathrm{K}} Z^{\mathrm{K}} - \dot{Z}^{\mathrm{K}} Y^{\mathrm{K}}}{Z_{\mathrm{K}}^2}. \tag{2.13}$$

Durch Einsetzen dieses Ergebnisses in die Bewegungsgleichungen (2.11) und den Zusammenhang $X^{\mathrm{K}} = x_1^{\mathrm{B}} Z^{\mathrm{K}}$ bzw. $Y^{\mathrm{K}} = y_1^{\mathrm{B}} Z^{\mathrm{K}}$ ergibt sich

$$\dot{x}_1^{\mathrm{B}} = \frac{1}{Z^{\mathrm{K}}} \left(v_x^{\mathrm{K}} - v_z^{\mathrm{K}} x_1^{\mathrm{B}} - \omega_x^{\mathrm{K}} Z^{\mathrm{K}} x_1^{\mathrm{B}} y_1^{\mathrm{B}} + \omega_y^{\mathrm{K}} Z^{\mathrm{K}} ((x_1^{\mathrm{B}})^2 - 1) - \omega_z^{\mathrm{K}} y_1^{\mathrm{B}} Z^{\mathrm{K}} \right) \tag{2.14}$$

und

$$\dot{y}_1^{\mathrm{B}} = \frac{1}{Z^{\mathrm{K}}} \left(v_y^{\mathrm{K}} - v_z^{\mathrm{K}} y_1^{\mathrm{B}} - \omega_x^{\mathrm{K}} Z^{\mathrm{K}} ((y_1^{\mathrm{B}})^2 + 1) + \omega_y^{\mathrm{K}} Z^{\mathrm{K}} x_1^{\mathrm{B}} y_1^{\mathrm{B}} + \omega_z^{\mathrm{K}} x_1^{\mathrm{B}} Z^{\mathrm{K}} \right). \tag{2.15}$$

Damit ergibt sich die Jacobi-Matrix für Punktmerkmale zu

$$\mathbf{J} = \begin{bmatrix} \frac{1}{Z^{\mathrm{K}}} & 0 & -\frac{x_1^{\mathrm{B}}}{Z^{\mathrm{K}}} & -x_1^{\mathrm{B}} y_1^{\mathrm{B}} & (x_1^{\mathrm{B}})^2 - 1 & -y_1^{\mathrm{B}} \\ 0 & \frac{1}{Z^{\mathrm{K}}} & -\frac{y_1^{\mathrm{B}}}{Z^{\mathrm{K}}} & -1 - (y_1^{\mathrm{B}})^2 & x_1^{\mathrm{B}} y_1^{\mathrm{B}} & x_1^{\mathrm{B}} \end{bmatrix}. \tag{2.16}$$

Basierend darauf können komplexere Bildmerkmale entworfen werden [173].

Da die bildbasierte Regelung aus mehreren Schritten besteht und in jedem Schritt durch die Jacobi-Matrix eine Bewegung impliziert wird, ändern sich die Merkmale und somit auch die Jacobi-Matrix in jedem Schritt. Die Merkmale $\mathbf{m}(t)$ und die Jacobi-Matrix $\mathbf{J}(t)$ sind somit zeitabhängig und müssen in jedem Schritt geschätzt werden.

Unter Annahme eines exponentiell abnehmenden Fehlers $\mathbf{e}_{\mathrm{m}}$ ergibt sich $\dot{\mathbf{e}}_{\mathrm{m}} = -\lambda_{\mathrm{e}} \mathbf{e}_{\mathrm{m}}$. Gemäß [17] lautet mit Gl. (2.10) der Regelschritt

$$\begin{bmatrix} \mathbf{v}^{\mathrm{K}}(t) \\ \boldsymbol{\omega}^{\mathrm{K}}(t) \end{bmatrix} = -\lambda_{\mathrm{e}} \hat{\mathbf{J}}^{+}(t) \mathbf{e}_{\mathrm{m}}(t), \tag{2.17}$$

wobei mit $\hat{\mathbf{J}}^+(t)$ die Pseudoinverse der geschätzten Jacobi-Matrix bezeichnet wird. Mit λ_e ist die Geschwindigkeit des Reglers einstellbar. Die Darstellung von $\boldsymbol{\lambda}_e$ als Vektor ermöglicht eine unterschiedliche Sensitivität der Merkmale.

Potentielle Probleme bei der bildbasierten Regelung sind Singularitäten der geschätzten Bild-Jacobi-Matrix $\hat{\mathbf{J}}(t)$ und lokale Minima, die sich aus $\mathbf{m} - \mathbf{m}^* \in \text{kern}(\hat{\mathbf{J}})$ ergeben [16]. Somit ist für einen Regler mit gutartigen Konvergenzeigenschaften eine hinreichend genaue Bestimmung und Schätzung der Bildmerkmale und der Jacobi-Matrix erforderlich, während Fehler bei der Kamera- und bei der *eye-in-hand*-Kalibrierung einen verschwindend geringen Einfluss haben [41]. Im Allgemeinen ist der Regler gemäß [71] asymptotisch stabil, falls $\hat{\mathbf{J}}^+(t)\mathbf{J}(t) > 0,\ \forall t$. Dies ist insbesondere dann der Fall, wenn die Jacobi-Matrix hinreichend genau geschätzt wird und $\hat{\mathbf{J}}^+$ wie auch $\mathbf{J}$ den vollen Rang 6 besitzen bzw. die jeweiligen Merkmale voneinander entkoppelt werden können [17].

Modernere Verfahren der bildbasierten Regelung enthalten darüber hinaus den Umgang mit Nebenbedingungen [1, 27] und eine Kollisionsvermeidung [132, 166].

3 Methoden für effiziente CNNs

Heutzutage stellen CNNs das Standardverfahren zur Klassifikation und Detektion für Bilddaten dar. Ihre Verwendung im industriellen Kontext stellt aber eine Besonderheit dar, da die entsprechenden Bilddaten deutlich weniger komplex sind als die üblichen Vergleichsdatensätze wie *ImageNet* [32] und andererseits spezielle Anforderungen an die Netze gestellt werden, welche sich aus maximalen Laufzeiten und Beschränkungen der genutzten Hardware ergeben. Gleichzeitig wird eine Klassifikationsgenauigkeit gefordert, die deutlich über der der klassischen Verfahren liegt, um die Verwendung des *Black-Box*-Modells CNN zu rechtfertigen.

In diesem Kapitel werden Methoden für die Generierung effizienter neuronaler Netze für Bildklassifikationsaufgaben beschrieben, die im Rahmen dieser Arbeit entstanden sind. Effizienz korrespondiert in diesem Sinne mit der Testgenauigkeit der CNNs bzgl. der Klassifikationsaufgabe und den benötigten Ressourcen für die Inferenz. Die Methoden dieses Kapitels sind kombinierbar und lassen sich in vier thematische Bereiche gliedern. Die vier Bereiche werden hier kurz vorgestellt. Am Ende dieses Kapitels wird in Abschnitt 3.6 die vollständige Verarbeitungskette für effiziente CNNs präsentiert, welche Verfahren aller Bereiche kombiniert.

Der erste Bereich umfasst den Aufbau und die Definition eines Trainingsdurchlaufs und seine Implikationen für die Hyperparameter. In Abschnitt 3.1 wird dazu insbesondere der Einfluss der Chargengröße auf das Training mit Augmentierung untersucht, da durch Augmentierung der Datensatz potentiell unendlich groß ist und die herkömmliche Definition eines Trainingsdurchlaufs daher weniger sinnvoll erscheint.

Im zweiten Bereich werden Methoden zur Konstruktion invarianter Merkmale beschrieben. Gemäß [137] existieren in der Praxis hierzu zwei Wege. Beim normalisierenden Weg wird eine deterministische Transformation auf ein Bild ausgeführt, um den Raum der unterschiedlichen

Ausprägungsformen eines Merkmals zu reduzieren. Beispielsweise hat das Merkmal *Kontrast* in Bildern eine kleinere Menge an möglichen Ausprägungen, wenn auf die Bilder zuvor ein Histogrammspreizungsverfahren angewendet wird. Durch normalisierende Bildverbesserungs- bzw. Vorverarbeitungsverfahren wird das Gesamtsystem invariant gegenüber bestimmten Merkmalen. Ein Verfahren zum Finden einer geeigneten Vorverarbeitungsstrategie wird in Abschnitt 3.2 vorgestellt. Im Gegensatz dazu werden beim integrativen Weg alle darstellbaren Erscheinungsformen eines Merkmals beim Entwurf des Klassifikators berücksichtigt, um invariant gegenüber dieses Merkmal zu werden. Angewandt auf CNNs entspricht dies der Augmentierung in der Bilddomäne. Hierbei werden die Eingabebilder hinsichtlich bestimmter Merkmale stochastisch variiert. Beispiele für diese Variationen sind die Rotation des Bildes um einen zufälligen Winkel oder eine zufällige Änderung der mittleren Helligkeit des Bildes. Das Finden einer geeigneten Augmentierungsstrategie ist Thema von Abschnitt 3.3. Prinzipiell können für die Bildverbesserungsverfahren und Augmentierung die gleichen Verfahren eingesetzt werden, jedoch sind die Parameter dieser Verfahren bei Augmentierung stochastisch, während für die Bildverbesserung ein Verfahren mit deterministischen Parametern auf jedes Bild, das in das CNN eingegeben wird, angewandt wird.

Der dritte Bereich beschreibt die Quantisierung von Gewichten und Aktivierungen eines CNN und wird in Abschnitt 3.4 thematisiert. Üblicherweise wird die Inferenz eines CNN mit einfacher (32-Bit-Gleitkomma-Arithmetik) oder halber Genauigkeit (16-Bit-Gleitkomma-Arithmetik) ausgeführt. Frühere Publikationen deuten darauf hin, dass ein CNN während des Trainings eine Vielzahl an Redundanzen aufbaut, die anschließend wieder entfernt werden können [60, 64, 114]. Das Ziel bei der Quantisierung ist es, diese Redundanzen derart aufzulösen, dass eine bestimmte Darstellung der Gewichte möglich ist, die einen geringeren Speicherbedarf impliziert. Die Quantisierung hat i. A. einen regularisierenden Effekt, da der Lösungsraum beim Training reduziert wird bzw. die Gewichte nach dem Training mit Quantisierungsrauschen überlagert werden. Eine zu starke Regularisierung kann aber auch zu schlechteren Konvergenzeigenschaften des CNN führen.

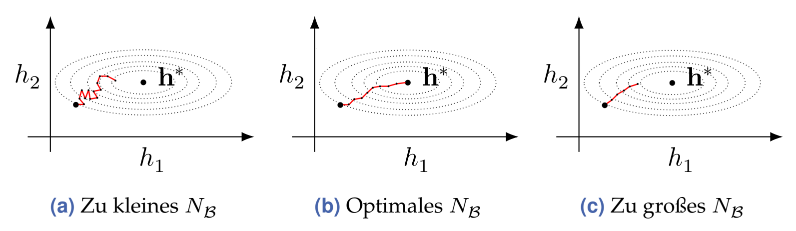
Die Suche nach einer geeigneten Topologie wird im vierten Bereich in Abschnitt 3.5 vor dem Hintergrund beschränkter Rechenressourcen behandelt. Die Suche nach der leistungsstärksten CNN-Topologie hat sich in der Forschung zum Entwurf immer größerer und komplexerer Netze hin entwickelt [58, 74, 131]. Dies wurde auch deshalb begünstigt, weil immer mehr annotierte Daten vorhanden sind und diese durch Verfahren der Domänenrandomisierung [163] beliebig vergrößert werden können. Da industrielle Anwendungen meist weniger komplex sind, sind jedoch kleinere CNNs oft schon ausreichend. In diesem Bereich wird daher der Frage nachgegangen, wie ein solches kleineres CNN entworfen werden kann.

3.1 Einfluss der Chargengröße bei der Augmentierung

Wie in Abschnitt 2.2.3 bereits aufgeführt, führen große Chargengrößen $N_{\mathcal{B}}$ zu einem schlechter generalisierenden CNN. Als Ursache hierfür wird in [84] Überanpassung, Anziehung zu Sattelpunkten, Fehlen explorativer Eigenschaften und die Konvergenz zu verschiedenen Optima genannt, während der Effekt nach [50] auf das niedrigere Rauschen bei der Schätzung des Gradienten zurückzuführen ist.

Einerseits bleibt der Effekt unbeachtet, dass bei größeren $N_{\mathcal{B}}$ weniger Iterationen beim Training durchgeführt werden, was eine langsamere Konvergenz bedingt. Andererseits führen größere Chargen aufgrund der Konsistenz des Optimierers zu einer rauschärmeren und somit zu einer genaueren Schätzung des Gradienten. Diese Problematik ist in Abb. 3.1 illustriert.

Zwar reduziert eine Skalierung des Gradientenschritts gemäß [90] den destruktiven Effekt großer Chargengrößen, jedoch legen einige Publikationen nahe, dass im Vergleich zu einer kleineren Chargengröße trotz dessen ein Verlust an Testgenauigkeit auftritt [50, 68, 113, 117]. Dennoch werden, um den Stand der Technik für Referenzdatensätze wie CIFAR-10 zu erreichen, anstelle einer Erhöhung der Iterationen eine Erhöhung der Durchläufe und eine Vergrößerung des CNN durchgeführt. Das Training wird hierbei durch eine größere Chargengröße beschleunigt [30, 54, 117, 175].



(a) Zu kleines $N_{\mathcal{B}}$ **(b)** Optimales $N_{\mathcal{B}}$ **(c)** Zu großes $N_{\mathcal{B}}$

Abbildung 3.1 Problem der Chargengröße $N_{\mathcal{B}}$: Zu kleine Chargengrößen führen zu verrauschten Iterationen (a), während zu große Chargengrößen nur langsam konvergieren (c).

Um die Vorteile von sowohl kleinen als auch großen Chargengrößen zu erhalten, wird vorgeschlagen, die Anzahl an Iterationen pro Durchlauf unabhängig von der Chargengröße, die möglichst groß anzusetzen ist, zu wählen [68, 84, 126]. In Abschnitt 4.2 wird untersucht, welche Implikationen dieser Vorschlag für die Chargengröße in Kombination mit einer Augmentierung im Bildbereich hat, da hierdurch eine unbegrenzte Anzahl an Bildern erzeugt werden kann, was die Festsetzung der Anzahl an Iterationen abhängig von der Anzahl der Bilder obsolet macht.

3.2 Vorverarbeitung

Der Inhalt des folgenden Abschnitts wurde erstmals in [188] veröffentlicht und beschreibt den normalisierenden Weg zum Entwurf invarianter Merkmale nach [137]. Nachdem der Stand der Technik in Abschnitt 3.2.1 beschrieben wird, folgt in Abschnitt 3.2.2 der eigene Beitrag zur automatischen Suche nach einer optimalen Vorverarbeitung. Hierzu werden die Bildverbesserungsverfahren aus Abschnitt 2.1 explizit zur Vermeidung bestimmter Variationen von Bildmerkmalen verwenden, indem die zugehörigen verarbeiteten Bilder gewichtet und aufsummiert werden. Die klassischen Bildverbesserungsverfahren haben gegenüber der direkten Vorverarbeitung mit CNNs den Vorteil, dass sie effizient auf der CPU berechnet werden können und dass CNNs diese Methoden im Allgemeinen nicht darstellen können.

3.2.1 Stand der Technik

CNNs werden ihrerseits verwendet, um Bilder zu verbessern, bspw. für Unterwasserbilder [99], Infrarotbilder [23, 96] oder um die Wahrnehmung des menschlichen Auges zu verbessern [158]. Diese Methoden haben allerdings den Nachteil, dass sie rechenintensiv und kaum interpretierbar sind.

In [22] werden die resultierenden Bilder dreier verschiedener Bildverbesserungsverfahren mit einem CNN fusioniert, um ein möglichst gut ausgeleuchtetes Bild zu erhalten. Jedoch stellt die Untersuchung des CNNs bzgl. besserer Generalisierung Γ bzw. Testgenauigkeit beim Training mit den verbesserten Bildern eine wenig beachtete Angelegenheit dar, da meist nur wenige Methoden betrachtet werden.

Radarbilder mit synthetischer Apertur (SAR) haben meist eine schlechte Qualität aufgrund schlechten Kontrastes, einer schlechten Beleuchtung, Atmosphären- und Sensorrauschen und einer geringen Auflösung des verwendeten Radargeräts. In [148] werden Verfahren der Rauschentfernung und zur Verbesserung der Auflösung auf SAR-Bilder angewandt, um die Klassifikationsleistung einer Support-Vektor-Maschine zu verbessern. Ein ähnliches Verfahren wird in [15] für die Vorverarbeitung von Röntgenbildern von Händen verwendet. Die Genauigkeit eines CNN, das mit diesen Bildern trainiert wird, steigt.

In [55] wird Bildverbesserung zur Steigerung der Erkennbarkeit von Augenkrankheiten verwendet. Durch die Subtraktion des mittleren Farbwerts, die Neuskalierung und das Zuschneiden des Bildes werden Schwankungen der Beleuchtung und der Kameraauflösung reduziert. Eine andere Arbeit zeigt, dass Vorverarbeitungsschritte wie Subtraktion des mittleren Farbwerts, varianzbasierte Standardisierung und Nullkomponentenanalyse die Klassifikationsgenauigkeit verschiedener kleiner CNNs erhöhen können [122].

Zusätzlich zur Topologie werden in [117] auch verschiedene Vorverarbeitungsschritte untersucht. Neben der Skalierung und dem zufälligen Zuschneiden führt auch die Verwendung einer Transformation mit einer (1×1)-Faltung zu einer Erhöhung der Genauigkeit des CNN. Außerdem zeigt sich, dass der RGB-Farbraum den anderen Farbräumen einschließlich verschiedener Grauwerträume überlegen ist. Methoden wie lokale Histogrammspreizung resultieren dagegen in einer Verschlechterung. In

den Versuchen aus [125] erweist sich ebenfalls der RGB-Farbraum den anderen Farbräumen als überlegen.

In [21] wird Vorverarbeitung explizit als Normalisierung verwendet. Hierbei werden handgeschriebene Ziffern bzgl. verschiedenster Schreibgewohnheiten wie Winkel, Position, Größe und Dicke normalisiert, um die Variationen der individuellen Zeichen zu reduzieren. Es wird gezeigt, dass dadurch die Klassifikationsgenauigkeit ungesehener Ziffern deutlich erhöht wird.

In [123] werden verschiedene Bildverbesserungsverfahren zum Erkennen von Gesichtsausdrücken mittels CNNs verwendet. Die untersuchten Methoden werden separat ausgeführt und die jeweiligen Ergebnisse werden miteinander verglichen.

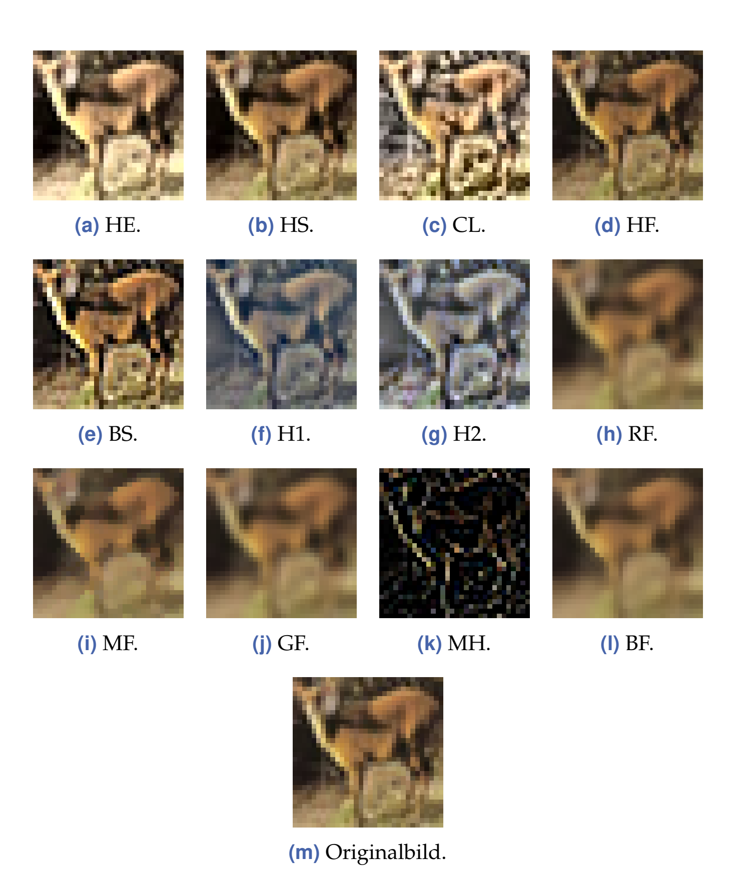
Diese Beispiele zeigen einerseits, dass durch Vorverarbeitung eine Verbesserung der Generalisierung Γ möglich ist, aber weisen andererseits darauf hin, dass die Auswahl der Methoden Expertenwissen erfordert, da ansonsten Verschlechterungen eintreten. Bis auf wenige rudimentäre Ansätze ist zum Zeitpunkt des Schreibens dieser Arbeit kein Verfahren bekannt, das die Vorverarbeitungsmethoden zu einem bestimmten Datensatz automatisch ermittelt und kombiniert. Eine solche Kombination birgt jedoch großes Potential in sich.

3.2.2 Aufgabenspezifische Vorverarbeitung

Im Folgenden wird ein Verfahren zur automatischen Suche nach einer Vorverarbeitungs- bzw. Bildverbesserungsstrategie (aSV) für die Eingabebilder von neuronalen Netzen vorgestellt. Die Methode nimmt an, dass die Lern- und Testdaten aus der gleichen Domäne stammen und sich hinreichend gut ähneln. Die $N_{\mathrm{V}} = 12$ verwendeten Verbesserungsverfahren sind Histogrammspreizung (HS), Histogrammegalisierung (HE), CLAHE (CL), Homogenisierung erster (H1) und zweiter (H2) Ordnung, Rechteckfilterung (RF), bilaterale Filterung (BF), Filterung mit dem Marr-Hildreth-Operator (MH), Homomorphe Filterung (HF), Bildverschärfung (BS), Gaußfilterung (GF) sowie Medianfilterung (MF) und werden in Abschnitt 2.1 beschrieben. Die resultierenden Bilder sind für ein Beispiel in Abb. 3.2 dargestellt. Jede Methode kann durch eine Funktion $F_{\mathrm{V},i}(\mathbf{I};\mathbf{a}_{\mathrm{V},i})$ dargestellt werden, wobei $\mathbf{a}_i$ für die Parameter der jeweiligen Bildverbesserungsmethode steht. Die Funktion $F_{\mathrm{V},0}$ ist als Identität

definiert, um zusätzlich das Originalbild zu verwenden. Das aus der Vorverarbeitung resultierende Gesamtbild $\mathbf{I}_\mathrm{V}$ ergibt sich aus der gewichteten Summe der verarbeiteten Bilder und des Originalbildes. Es gilt

$$\mathbf{I}_\mathrm{V} = \sum_{i=0}^{N_\mathrm{v}} \theta_{\mathrm{V},i} \; F_{\mathrm{V},i}(\mathbf{I}; \mathbf{a}_{\mathrm{V},i}) \text{ mit } \sum_{i=0}^{N_\mathrm{v}} \theta_{\mathrm{V},i} = 1. \tag{3.1}$$



(a) HE. (b) HS. (c) CL. (d) HF.
(e) BS. (f) H1. (g) H2. (h) RF.
(i) MF. (j) GF. (k) MH. (l) BF.
(m) Originalbild.

Abbildung 3.2 Verwendete Bildverbesserungsverfahren für die aufgabenspezifische Vorverarbeitung nach Abschnitt 3.2.2 am Beispiel eines Bildes des CIFAR-10-Datensatzes.

Da die Methoden i. A. hochgradig nichtlinear bzgl. $\mathbf{a}_{\mathrm{V},i}$ sind, eignen sie sich nicht, um während des Trainings eines CNN bestimmt zu werden.

Daher werden die Operationen mithilfe eines Mischfaktors $\gamma_{V,i}$ zwischen dem Originalbild **I** und dem mit einem geeigneten Satz an Parametern $\mathbf{a}_{V,i}^{\max}$ verarbeiteten Bild linearisiert. Somit ergibt sich

$$F_{V,i}(\mathbf{I}; \mathbf{a}_{V,i}) \approx (1 - \gamma_{V,i})\mathbf{I} + \gamma_{V,i} F_{V,i}(\mathbf{I}; \mathbf{a}_{V,i}^{\max}) =: G_{V,i}(\mathbf{I}, \gamma_{V,i}). \quad (3.2)$$

Der Wertebereich des resultierenden Bildes wird anschließend an den des Originalbildes angepasst. Das Verfahren ist in Abb. 3.3 illustriert.

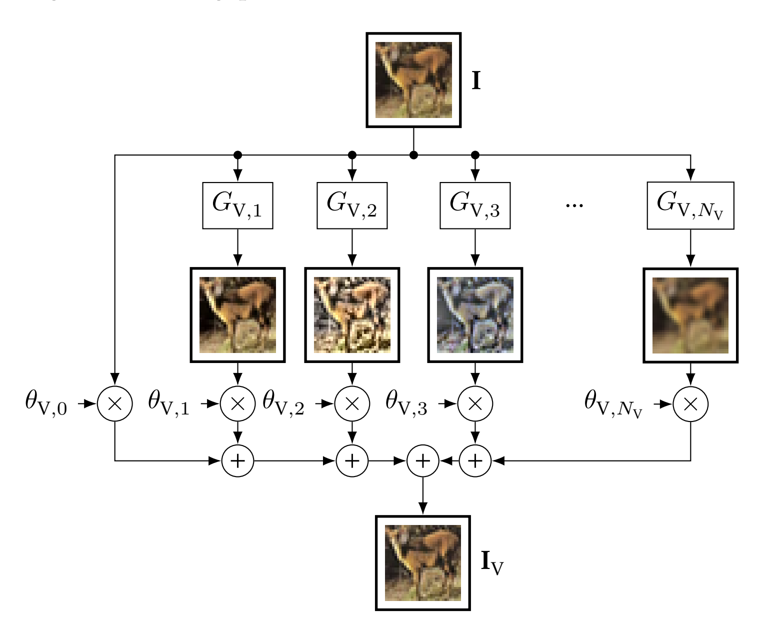


Abbildung 3.3 Schematischer Ablauf des Verfahrens der aufgabenspezifischen Vorverarbeitung.

Somit ist für das Verfahren lediglich die Bestimmung der Gewichte $\theta_{V,i}$ und der Linearisierungsfaktoren $\gamma_{V,i}$ erforderlich. Hierzu wird der Merkmalsextraktor eines vortrainierten CNN verwendet. Die Gewichte des Merkmalsextraktors sind dabei nicht trainierbar. Lediglich die letzte vollständig verbundene Schicht und die Schicht zur Bestimmung von $\theta_{V,i}$ und $\gamma_{V,i}$ sind trainierbar. Somit wird die Parameterkombination gesucht, für die der Informationsgehalt bzgl. der Klassifikationsaufgabe maximal

wird bzw. die Irrelevanz minimal wird. Die Bedingung $\sum_{i=0}^{N_\text{v}} \theta_{\text{V},i} = 1$ wird über eine *softmax*-Funktion sichergestellt.

Mithilfe der gefundenen Parameter können anschließend die korrespondierenden Datensätze verarbeitet und als Eingabe in das zu trainierende finale Netz verwendet werden.

Vorverarbeitungsverfahren mit *Straight-Through*-Schätzer

Im Folgenden wird eine Verbesserung des aSV-Verfahrens mittels *Straight-Through*-Schätzer (aSVST) beschrieben, die nicht in [188] veröffentlicht wurde. Anstatt der Verwendung einer Linearisierung zwischen vorverarbeitetem Bild und Originalbild mittels $\gamma_{\text{V},i}$ wird ein *Straight-Through*-Schätzer gemäß [8] wie in [61] verwendet (vgl. Abschnitt 3.3.1). Dieser erlaubt die approximative Berechnung des Gradienten bzgl. jeweils eines Parameters a_i. Für die Bildverbesserungsverfahren HS, HE, RF, MF und MH wird kein Parameter verwendet, da diese Operationen parameterfrei sind bzw. der Filterkern mit der Größe 3×3 konstant ist. Bei CL wird die Kachelgittergröße (engl. *tile grid size*) mit $\lfloor 16 - 8 \cdot a_{\text{V},i} \rfloor$ und bei BF werden beide Gaußkerne mit demselben Parameter parametrisiert. Bei allen anderen Verfahren wird der Gradient bzgl. der Standardabweichung des Gaußkerns approximativ gebildet. Die Größe der Filterkerne ist fest und beträgt gemäß den Ergebnissen aus Abschnitt 4.3.1 3×3 bzw. 21×21. Die Ermittlung der Parameter erfolgt genauso wie im ursprünglichen Verfahren (s. o.).

Approximation des Vorverarbeitungsverfahrens

Ebenfalls nicht in [188] veröffentlicht wurde die im Folgenden beschriebene Approximation des aSV- bzw. aSVST-Verfahrens. Für die Inferenz ist es meist nicht praktikabel, alle N_V Vorverarbeitungsverfahren für jedes Eingabebild auszuführen, da dies den Ressourcenbedarf oder die Rechendauer erhöht. Die Ergebnisse des aSV- bzw. aSVST-Verfahrens (s. 4.3.1 bzw. [188]) zeigen allerdings, dass bereits wenige Vorverarbeitungsmethoden plus dem Originalbild den überwiegenden Anteil an der Kombination stellen.

Für die Approximation werden zunächst alle Anteile der Bildverbesserungsverfahren, für deren Mischfaktor $\gamma_{\text{V},i} < 0{,}05$ gilt, zu $\theta_{\text{V},i} = 0$ gesetzt. Anschließend werden die drei Verfahren mit dem höchsten Anteil $\theta_{\text{V},i}$

ausgewählt und der Anteil aller anderen Verfahren wird ebenfalls zu $\theta_{\mathrm{V},i} = 0$. Die Summe der Anteile der entfernten Verfahren wird dem des Originalbilds $\theta_{\mathrm{V},0}$ hinzugefügt. Somit besteht die Approximation aus den drei Methoden mit den höchsten Anteilen und dem Originalbild.

3.3 Augmentierungsstrategien

Für das Augmentieren von Bildtrainingsdaten hat sich eine reine Augmentierung auf den Bilddaten durchgesetzt, da diese intuitiv und durch die Arbeiten von Schulz-Mirbach [137] interpretierbar ist. Die Bildverarbeitung bietet ein weites Repertoire an möglichen Operationen für eine Augmentierung. Diese umfassen Koordinatentransformationen, Farbwertmanipulationen, Entfernen von Informationen und das Hinzufügen von Rauschen. Jedoch ist das Verhalten einzelner Operationen für die Augmentierung datensatzspezifisch und kann daher positive wie negative Einflüsse auf die Testgenauigkeit haben [67]. Im Folgenden werden daher Verfahren für die automatische datensatzspezifische Suche nach einer geeigneten Augmentierungsstrategie vorgestellt, die auf Grundlage des Lerndatensatzes $\mathcal{X}_{\mathrm{L}}$ ermittelt werden. Es wird implizit angenommen, dass die Lern- und Testdaten mit der gleichen Verteilung erzeugt wurden und sich somit hinreichend ähneln.

3.3.1 Stand der Technik

AutoAugment [30] ist die erste Veröffentlichung zur systematischen Suche nach einer optimalen Augmentierungsstrategie für Objektklassifikationsaufgaben. Hierfür wird der Suchraum für die Augmentierung diskretisiert, indem die Augmentierungsstrategie $\mathcal{T}_{\mathrm{aug}}$ in 25 Unterstrategien unterteilt wird, die durch ein Tupel von zwei Operationen beschrieben werden. Jede Operation $O_{i,\mathrm{AA}}$ eines Tupels wird durch zwei Parameter beschrieben: die Wahrscheinlichkeit $p_{i,\mathrm{AA}}$, dass die Operation ausgeführt wird, und, falls diese ausgeführt wird, die Stärke $\mu_{i,\mathrm{AA}}$ der Operation. Somit können durch eine Unterstrategie vier verschiedene Bilder erzeugt werden. Bei 25 Unterstrategien ergeben sich somit aufgrund der redundanten Identität maximal 76 mögliche Ausgaben für ein Bild. Dies entspricht in etwa der Anzahl, die notwendig ist, damit sich Offline- und

Online-Augmentierung ähnlich verhalten (s. Anhang A.1). Als Operationen werden die Funktionen der `PIL`-Bibliothek verwendet, die ein Bild als Eingabe akzeptieren, sowie *Cutout* [35] und *SamplePairing* [76]. Insgesamt sind dies 16 Operationen, wobei die Identität nicht explizit als Operation aufgeführt wird. Die beiden Parameter einer Operation werden jeweils auf elf Werte diskretisiert. Die beiden Operationen einer Unterstrategie werden in einer Kaskade zusammen mit den drei Operationen zufälliger Spiegelung, zufälligem Ausschneiden und *Cutout* ausgeführt, sodass bis zu fünf Operationen auf ein Bild ausgeführt werden. Für die diskrete Suche wird Verstärkungslernen verwendet. Dabei wird in jedem Schritt ein CNN mit der aktuellen Strategie trainiert und die Generalisierungsfähigkeit des CNN anhand von 15.000 Stichproben der Strategie gemessen. Anhand dessen wird die Strategie anschließend aktualisiert. Das Verfahren erreicht zwar auf den verwendeten Datensätzen sehr hohe Genauigkeiten, jedoch sind die hierfür notwendigen Rechendauern von über 1.000 GPU-Stunden nicht praktikabel.

Lim et al. [102] beschleunigen mit *fast AutoAugment* die Suche nach der Augmentierungsstrategie $\mathcal{T}_{\mathrm{aug}}$, indem Bayes'sche Optimierung verwendet wird, was die Anzahl an zu evaluierenden Augmentierungsstrategien stark reduziert. Darüber hinaus wird anstelle des Trainings eines CNN und der Bestimmung der Fähigkeit zur Generalisierung die Ähnlichkeit der Verteilungen der Datensätze gemessen (engl. *density matching*). Dies wird dadurch motiviert, dass der augmentierte Lerndatensatz durch seinen größeren Stichprobenumfang bei geeigneter Augmentierung dem Testdatensatz bzw. ungesehenen Daten bzgl. seiner intrinsischen Verteilung ähnlicher wird. Da Testdaten i. A. während des Trainings nicht verfügbar sind, wird der Lerndatensatz $\mathcal{X}_{\mathrm{L}}$ in einen zu augmentierenden Teildatensatz $\mathcal{X}_{\mathrm{A}}$ und einen Validierungsdatensatz $\mathcal{X}_{\mathrm{V}}$ unterteilt. Der Vergleich der Verteilungsfunktionen geschieht indirekt, indem ein Modell mit $\mathcal{X}_{\mathrm{V}}$ trainiert wird und die Genauigkeit von $\mathcal{T}_{\mathrm{aug}}(\mathcal{X}_{\mathrm{A}})$ an diesem Modell ermittelt wird. Jede Unterstrategie wird so bei einer unterschiedlichen Aufteilung des Lerndatensatzes $\mathcal{X}_{\mathrm{L}}$ unabhängig voneinander bestimmt. *Fast AutoAugment* reduziert die Rechendauer circa um den Faktor 1.000, führt aber zu leicht schlechteren Ergebnissen.

In *RandAugment* [31] wird kein Modell mehr verwendet, um die Qualität der Augmentierung zu bestimmen, da dies zu suboptimalen Er-

gebnissen führt. Stattdessen wird der Suchraum verkleinert, indem jede Operation mit der gleichen Wahrscheinlichkeit ausgewählt wird. Somit sind statt bestimmter Unterstrategien beliebige Kombinationen der Augmentierungsoperationen möglich. Der Stärkeparameter ist für alle Operationen gleich. Somit sind mit der Länge der Kaskade und dem Stärkeparameter lediglich zwei Parameter für $\mathcal{T}_{\text{aug}}$ zu bestimmen. Nach einer Diskretisierung in jeweils zehn Werte hat der Suchraum eine Größe von 100. Dies reicht aus, um die Augmentierungsstrategie direkt bzgl. des Ziel-CNN zu optimieren.

Der Ansatz von *faster AutoAugment* (FAA) [61] verwendet eine GAN-artige Struktur, um zwischen augmentierten und unveränderten Daten zu unterscheiden. Dabei werden die gemäß der gefundenen Augmentierungsstrategie $\mathcal{T}_{\text{aug}}$ augmentierten Bilder $\mathcal{T}_{\text{aug}}(\mathcal{X}_{\mathrm{A}})$ anstelle der von durch ein Generatornetzwerk generierten Bilder verwendet. Bei der Augmentierungsstrategie wird die Einteilung in Unterstrategien gemäß dem *AutoAugment*-Verfahren verwendet und die Strategie wird während der Lernphase des Diskriminators erlernt, welcher hinsichtlich der Wasserstein-Distanz und der Klassifikationsfähigkeit zwischen Original- und augmentierten Daten optimiert wird. Für das Verfahren ist die Differenzierbarkeit der Augmentierungsstrategien sowie der jeweiligen Operationen bzgl. der Auswahlwahrscheinlichkeit und der Stärke der Operation notwendig. Als Verteilungsfunktion für die Ausführwahrscheinlichkeit $p_{i,\mathrm{AA}}$ wird daher eine kontinuierliche Bernoulli-Verteilung nach [80] verwendet. Für die Stärke $\mu_{i,\mathrm{AA}}$ einer Operation wird ein *Straight-Through*-Schätzer gemäß [8] verwendet. Dieser definiert die Operation zu $\hat{O}_i(\mathbf{I}, \mu_{i,\mathrm{AA}}) = \mathrm{StopGradient}(O_{i,\mathrm{AA}}(\mathbf{I}, \mu_{i,\mathrm{AA}}) - \mu_{i,\mathrm{AA}}) + \mu_{i,\mathrm{AA}}$, wodurch sie differenzierbar bzgl. $\mu_{i,\mathrm{AA}}$ wird, da mithilfe der Funktion $\mathrm{StopGradient}$ der Gradient der ursprünglichen Operation $O_{i,\mathrm{AA}}$ unterdrückt wird. Für die Suche nach den Operationen einer Unterstrategie wird ein von [107] inspiriertes Verfahren verwendet, bei dem in der Lernphase eine gewichtete Summe der verwendeten Operationen berechnet wird, deren Gewichte trainierbar und aufgrund der *softmax*-Aktivierung nichtnegativ sind und sich auf 1 summieren. Nach dem Training werden die Gewichte als Koeffizienten einer verallgemeinerten Bernoulli-Verteilung interpretiert, anhand derer die Unterstrategien abgetastet werden.

Im Gegensatz dazu generieren andere Ansätze künstlich Daten mithilfe von Bayes'schen Ansätzen [162], konventionellen GAN-Strukturen [83] oder Domänenrandomisierung [163]. Dies zielt darauf ab, die Verteilung der augmentierten Lerndaten zu erweitern, anstatt sie zu interpolieren, mit der Absicht, dass die Verteilung der Testdaten innerhalb der vergrößerten Verteilung der augmentierten Daten liegt und nicht mehr zu dieser möglichst ähnlich ist. Dieser Ansatz wird in diesem Teil der Arbeit nicht weiter betrachtet, da eine solche Vergrößerung eine höhere Kapazität des CNN erfordert, was zu Ineffizienzen führen kann.

3.3.2 Modifiziertes AutoAugment-Verfahren

Im Folgenden wird das FAA-Verfahren aufgegriffen und für die Zwecke dieser Arbeit weiterentwickelt.

Die Ursache für die hohe Laufzeit liegt in der Größe des Suchraums für die *AutoAugment*-Verfahren. Bei *RandAugment* wird der Suchraum zwar enorm reduziert, jedoch hat dies zur Folge, dass für alle Operationen die Stärke μ_{AA} verwendet wird. Dies wird der Tatsache nicht gerecht, dass je nach Datensatzdomäne CNNs unterschiedlich sensibel gegenüber bestimmten Varianzen sein sollten. Bspw. ist bei Bildern mit bestimmten Texturen in den Testdaten eine höhere Varianz bzgl. Rotation zu erwarten als bzgl. der Helligkeit, wenn alle Bilddaten unter den gleichen Umgebungsbedingungen aufgenommen wurden. Ferner ist *RandAugment* dahingehend problematisch, dass ungewollte Kombinationen von Operationen auftreten können. Daher wird der Suchraum im Folgenden durch eine Kombination von *RandAugment* und *faster AutoAugment* reduziert: Die Struktur der Unterstrategien bleibt erhalten und die Ausführwahrscheinlichkeit $p_{i,\mathrm{AA}}$ wird entfernt. Die Stärke einer Operation folgt einer Gleichverteilung $\mathcal{U}(0, \mu_{i,\mathrm{AA}})$ bzw. $\mathcal{U}(-\mu_{i,\mathrm{AA}}, \mu_{i,\mathrm{AA}})$. Diese Formulierung erlaubt einerseits Operationen mit Vorzeichen (z. B. Winkel bei der Rotation, Reduktion und Erhöhung von Helligkeit, ...) und andererseits macht dies das FAA-Verfahren als Interpretation des integrativen Weges zur Konstruktion von invarianten Merkmalen nach Schulz-Mirbach [137] stringenter. Diese Methode wird im Weiteren als FAAR (*faster AutoAugment + Random Magnitude*) bezeichnet.

Gemäß [37] haben Unschärfe und Rauschen einen sehr schädlichen Effekt auf die Klassifikationsergebnisse eines trainierten CNN, falls kei-

ne Invarianzen gegenüber diesen Störungen aufgebaut wurden. Ferner mangelt es *AutoAugment* und dessen Erweiterungen an Operationen, die Irrelevanzen wie Rauschen hinzufügen oder Informationen entfernen. Daher werden das Hinzufügen von additivem normalverteiltem Rauschen, additivem gleichverteiltem Rauschen und Impulsrauschen sowie die Filterung mit einem Gaußtiefpass als weitere Operationen hinzugefügt. Zusätzlich werden durch die Operation *Farbverschiebung*, bei der ein konstanter Wert auf einen der drei Farbkanäle addiert wird, ein fehlender Weißabgleich und unterschiedliche Beleuchtungsquellen simuliert. Diese *AutoAugment*-Variante wird im Weiteren als FAARO (*faster AutoAugment + Random Magnitude + More Operations*) bezeichnet.

3.4 Quantisierung

Obwohl sich neuronale Netze als Stand der Technik für Bildverarbeitungsaufgaben in vielen Domänen durchgesetzt haben, werden die Netze i. d. R. mit mindestens halber Genauigkeit, d. h. mit 16-Bit-Gleitkomma-Arithmetik, entworfen, was deren Inferenz nur auf Geräten mit entsprechenden Kapazitäten erlaubt. Da nicht selten mehrere Millionen MAC-Operationen für die Inferenz nötig sind und eine ähnlich hohe Anzahl an Parametern gespeichert werden muss, sind die erforderlichen Ressourcen im Vergleich zu einem gröber quantisierten Netz um ein Vielfaches höher: Im Vergleich zur halben Genauigkeit benötigt eine 8-Bit-Festkomma-Architektur für eine MAC-Operation 18,5-mal weniger Energie, 27,5-mal weniger Platz auf einem Mikrochip und nur die Hälfte an Speicherplatz [70]. Eine solche Reduzierung der benötigten Ressourcen ermöglicht die Implementierung auf kleinen oder batteriebetriebenen Geräten wie herkömmlichen FPGAs oder Mobiltelefonen. Daher wird im Folgenden ein Quantisierungsverfahren für CNNs vorgestellt, das in [192] und dessen Erweiterungen in [185] veröffentlicht wurden.

3.4.1 Stand der Technik

Die bekannten Quantisierungsverfahren können im Wesentlichen in drei Gruppen eingeteilt werden: Die erste Gruppe enthält Methoden, die die

Gewichte eines CNN nach der Lernphase quantisieren, indem nach optimalen Quantisierungsstufen gesucht wird. Für diese Gruppe ist i. A. eine Umsetzungstabelle (engl. *lookup table*) in der Inferenz erforderlich, um die Originalgewichte aus den Quantisierungsstufen zurückzugewinnen. Da die MAC-Operationen dann mit halber Genauigkeit ausgeführt werden, wird durch diese Verfahren lediglich der Speicherbedarf der Inferenz reduziert. Chen et al. zeigen in [20], dass eine Kombination aus linearer Quantisierung und einer Quantisierung, deren Quantisierungsstufen der Häufigkeit der Werte der Gewichte folgen, den Varianten mit linear bzw. zufällig gewählten Quantisierungsstufen überlegen ist. Es können 6- bis 9-Bit-Repräsentationen eines VGG-Net erreicht werden, die einen Genauigkeitsverlust von ca. $1\,\%$ für CIFAR-10 erreichen. In [103] werden die optimalen Quantisierungsstufen für Gewichte und Aktivierungen analytisch berechnet und anschließend wird das CNN durch eine Feinabstimmung nachgelernt. Hierdurch wird für eine 8-Bit-Quantisierung bei CIFAR-10 eine Verbesserung der Genauigkeit von $0{,}03$ Prozentpunkten gegenüber der Gleitkomma-Arithmetik erreicht. In [57] wird das Verfahren *Ristretto* vorgestellt, das die Gleitkomma-Arithmetik eines CNN iterativ zu einer dynamischen Festkomma-Arithmetik überführt. Dabei werden die Histogramme der Gewichte und Aktivierungen analysiert und dann die am besten geeignete Bitlänge bestimmt. Falls keine Augmentierung verwendet wird, erreicht diese Methode einen Abfall an Genauigkeit von weniger als $0{,}9$ Prozentpunkten für CIFAR-10 für unterschiedliche CNN-Topologien. Das *Deep-Compression*-Verfahren nach [59, 60] unterteilt die Gewichte in k Cluster, was den Speicherbedarf auf $\log_2(k)$ Bit pro Gewicht reduziert. Anschließend wird mithilfe einer Huffman-Codierung der benötigte Speicherplatz weiter reduziert. Das Verfahren hat eine regularisierende Wirkung auf das CNN, da durch das Ausdünnen von großen CNNs bessere Optima gefunden werden können.

In der zweiten Gruppe werden Regularisierungsverfahren während der Lernphase verwendet, um den Quantisierungsfehler in der Testphase zu minimieren. Dadurch werden die Quantisierungseffekte schon während des Trainings berücksichtigt. Sheng et al. stellen in [140] einen Regularisierungsterm für MobileNets [72] vor, bei dem die Leistung des Quantisierungsrauschens der Aktivierungen und Gewichte bestraft wird.

Die Quantisierungsstufen sind somit bereits vor dem Training festgelegt und das CNN wird während des Trainings darauf optimiert. Jedoch führt dieses Verfahren zu einer Verschlechterung der Genauigkeit von über 1 % bei einer 8-Bit-Quantisierung, was auf die Quantisierung der BN-Schicht und die ReLU-Aktivierung, die in ein halboffenes Intervall abbildet, zurückzuführen ist.

Bei den Methoden der dritten Gruppe werden Skalierungsfaktoren während des Trainings verwendet, um den optimalen Wertebereich der Parameter zu finden. Bei der *dynamischen Festkomma-Arithmetik* nach [28] ist eine Bitlänge gegeben, für die ein Skalierungsfaktor adaptiv für jede Schicht des CNN berechnet wird, um die optimale Position des Kommas zu finden. Hierfür werden sog. Überlaufraten berechnet, die den Anteil der Parameter darstellen, die nicht durch den Wertebereich der Quantisierung darstellbar sind. Überschreitet diese Rate den Wert 0,01 Prozent, wird das Komma nach rechts verschoben. Ohne Augmentierung liefert das Verfahren für CIFAR-10 gegenüber halber Genauigkeit eine Verschlechterung der Testgenauigkeit von ca. 0,8 Prozentpunkten und benötigt eine 10-Bit-Quantisierung, wobei 5 Bit hinter dem Komma benötigt werden. Der Gewinn der dynamischen Festkomma-Arithmetik gegenüber der konventionellen Festkomma-Arithmetik beträgt hierbei 7 Bit. In [56] wird ein *Greedy*-Algorithmus verwendet, um die optimale Bitlänge zu bestimmen. Im Kontrast dazu stellt Jacob et al. in [78] eine Ganzzahl-Arithmetik für neuronale Netze vor, die auch Aktivierungen quantisiert. Während des Trainings werden hierfür 8-Bit-Ganzzahlen auf reelle Zahlen abgebildet, um eine Matrixmultiplikation zu einer Ganzzahlmultiplikation umzuwandeln. Dafür ist eine Skalierung mit einer reellen Zahl erforderlich, da die Gewichte auf Ganzzahlen im Bereich $[-128, 127]$ beschränkt sind.

3.4.2 Quantisierung mittels Skalierungsfaktoren

Das im Folgenden beschriebene Verfahren ist in die dritte Gruppe nach Abschnitt 3.4.1 einzuordnen. Für eine FPGA-Implementierung ist eine Quantisierung der Gewichte – insbesondere für BN-Schichten – und der Aktivierungen erforderlich. Dieses Verfahren vermeidet zusätzliche Operationen wie bspw. Skalieren während der Inferenz und verwendet ausschließlich Festkommazahlen derselben Länge für die Inferenz.

Ein Festkomma-Quantisierungsschema ist gegeben durch die Ganzzahlbitlänge $L_{\mathrm{q,I}}$, die Nachkomma-Bitlänge $L_{\mathrm{q,F}}$ und ggf. ein Vorzeichenbit. Da im Folgenden standardmäßig eine ReLU-Aktivierung verwendet wird, ist die Aktivierung nicht vorzeichenbehaftet. Somit wird nur für die Gewichte ein Vorzeichenbit benötigt, während dies für die Aktivierung nicht der Fall ist. Über die Bitlängen ist die Breite einer Quantisierungsstufe mit $\Delta_{\mathrm{q}} = 2^{-L_{\mathrm{q,F}}}$ und die maximale Amplitude mit $A_{\mathrm{q}} = 2^{L_{\mathrm{q,I}}}$ ableitbar. Für das im Folgenden beschriebene Verfahren ist die Festsetzung der Bitlängen für Aktivierungen $(L_{\mathrm{q,I,A}}, L_{\mathrm{q,F,A}})$ und Gewichte $(L_{\mathrm{q,I,W}}, L_{\mathrm{q,F,W}})$ vor Beginn des Trainings erforderlich. Zum Zwecke der einheitlicheren Darstellung wird das zusätzlich benötigte Vorzeichenbit der Gewichte als Teil der Ganzzahlbits $L_{\mathrm{q,I,W}}$ betrachtet. Die maximale Amplitude ist dann $A_{\mathrm{q,W}} = 2^{L_{\mathrm{q,I}}-1}$.

Bei der Verwendung einer (leaky) ReLU als Aktivierungsfunktion kann gezeigt werden, dass eine Skalierung der Ausgabekarte $\mathbf{Y}_l$ mit dem Skalierungsfaktor $s_l \geq 0$ gleichbedeutend ist mit der entsprechenden Skalierung der Filterkerne $\mathbf{k}_l$ und Biasterme b_l. Gl. (2.4) wird dann zu

$$s_l \cdot \mathbf{Y}_l = \sigma_{\mathrm{akt}}(\mathbf{X} ** (s_l \mathbf{k}_l) + s_l b_l). \tag{3.3}$$

Jeder Skalierungsfaktor s_l korrespondiert mit genau einem Filterkern $\mathbf{k}_l$. Das vorgestellte Quantisierungsverfahren nutzt diese Linearität, um für jeden Filterkern einen Skalierungsfaktor zu bestimmen, sodass der durch $L_{\mathrm{q,I,A}}$ und $L_{\mathrm{q,I,W}}$ vorgegebene Wertebereich eingehalten wird. Daraus ergeben sich die folgenden Bedingungen an s_l:

$$\begin{aligned} Y_l^{\max} &:= \max\left\{\sigma_{\mathrm{akt}}(\mathbf{X} ** (s_l \mathbf{k}_l) + s_l b_l)\right\} \overset{!}{\leq} A_{\mathrm{q,A}} - \Delta_{\mathrm{q,A}}, \\ k_l^{\max} &:= \max\left\{|s_l \cdot \mathbf{k}_l|\right\} \overset{!}{\leq} A_{\mathrm{q,W}} - \Delta_{\mathrm{q,W}}, \\ |s_l \cdot b_l| &\overset{!}{\leq} A_{\mathrm{q,W}} - \Delta_{\mathrm{q,W}}. \end{aligned} \tag{3.4}$$

Für die Quantisierung des neuronalen Netzes werden die Skalierungsfaktoren für jede Faltungsschicht während der Lernphase durch einen gleitenden Mittelwert bestimmt. Die Skalierungsfaktoren werden nach

jeder Charge $\mathcal{B}$ aktualisiert. Für eine Faltungsschicht ergibt sich der Skalierungsfaktor s_l zu

$$s_l \leftarrow m_\mathrm{q} \cdot s_l + (1 - m_\mathrm{q}) \cdot z_{l,\mathcal{B}} \text{ mit}$$
$$z_{l,\mathcal{B}} = \min \left\{ \frac{\alpha_\mathrm{q} \cdot (A_\mathrm{q,A} - \Delta_\mathrm{q,A})}{Y_{l,\max}}; \frac{\alpha_\mathrm{q} \cdot (A_\mathrm{q,W} - \Delta_\mathrm{q,W})}{\max\{H_{l,\max}; b_l\}} \right\}, \tag{3.5}$$

wobei m_q das Momentum des gleitenden Mittelwertes ist, dessen Wert wie der der Chargennormalisierung bestimmt wird, und α_q einen Puffer darstellt. Der Puffer ist ein Hyperparameter, der den Überlauf einzelner Aktivierungen erlaubt oder bestraft. Die Skalierungsfaktoren werden auf einen Maximalwert begrenzt, um hohe Werte für den temporären Skalierungsfaktor $z_{l,\mathcal{B}}$ infolge von schwachen Aktivierungen zu vermeiden, und mit $s_l = 1$ initialisiert.

Da die Eingabe $\mathbf{X}$ in die Faltungsschicht a priori nicht bekannt ist, kann es dazu kommen, dass einzelne Werte der Ausgabekarte $\mathbf{Y}_l$ eines Filterkerns den maximal zulässigen Wert $A_\mathrm{q,A} - \Delta_\mathrm{q,A}$ überschreiten. In der Inferenz werden Aktivierungen, die diesen Wert überschreiten, abgeschnitten. Um den daraus folgenden negativen Effekt schon während des Trainings zu berücksichtigen, wird die ReLU wie folgt manipuliert. Außerdem wird die Rundung auf den Wert 0 bei der Quantisierung berücksichtigt, da diese insbesondere bei betragsmäßig kleinen Gewichten einen großen relativen Fehler verursacht. Die Aktivierung wird zu

$$\sigma_{\mathrm{q,\,akt}}(x) = \begin{cases} 0 & \text{für } x \leq \frac{\Delta_\mathrm{q,A}}{2}, \\ x & \text{für } \frac{\Delta_\mathrm{q,A}}{2} < x < A_\mathrm{q,A} - \Delta_\mathrm{q,A}, \\ \frac{x}{10} + \frac{9 \cdot (A_\mathrm{q,A} - \Delta_\mathrm{q,A})}{10} & \text{sonst.} \end{cases} \tag{3.6}$$

Um die Verwendung von vollständig verbundenen Schichten zu vermeiden, werden diese als Faltungsschicht mit (1×1)-Filterkernen interpretiert. Außerdem wird eine GAP-Schicht (engl. *global averaging pooling layer*) verwendet, was die Reduktion der Anzahl der (1×1)-Faltungsschichten auf eine einzige ermöglicht.

Die Skalierungsfaktoren für jeden Filterkern führen dazu, dass die Verteilung der Aktivierungen begrenzt ist. Dies kann als normalisierend betrachtet werden, wodurch der Einsatz einer Chargennormalisierungsschicht obsolet wird. Außerdem dient eine L_2-Regularisierung dazu,

Ausreißer bei den Gewichten zu vermeiden. Ferner wird durch eine zusätzliche L_1-Regularisierung eine höhere Anzahl an Nullen in den Gewichten erzwungen, was den mittleren Quantisierungsfehler reduziert.

Nach dem Training werden die Skalierungsfaktoren mit den jeweiligen Gewichten multipliziert. Man erhält $\mathbf{k}_l \leftarrow \mathbf{k}_l s_l$ und $b_l \leftarrow b_l s_l$. Die skalierten Gewichte werden dann quantisiert, indem die Werte stochastisch gerundet werden. Dies bedeutet, dass abhängig vom Abstand zu den nächsten beiden Quantisierungsstufen eine Wahrscheinlichkeit berechnet wird, mit der Wert zu der jeweiligen Stufe gerundet wird. Für die Inferenz können dann Faltungsschichten aus den Standardbibliotheken wie `tensorflow` verwendet werden. Um die Aktivierung gemäß dem Quantisierungsschema entsprechend zu runden und abzuschneiden, werden nach den Standardfaltungsschichten entsprechende Aktivierungsschichten eingefügt. Dieser Prozess wird im Folgenden als *Konvertierung* des quantisierten CNN bezeichnet.

3.4.3 Erweiterungen

Die Erweiterungen umfassen ein Vortraining zur Bestimmung geeigneter initialer Skalierungsfaktoren, ein Feinabstimmungsverfahren für die Gewichte, das *Channel-Pruning*-Verfahren nach He et al. [64] zur Reduktion von Redundanzen eines CNN und eine bisher unveröffentlichte Erweiterung, bei der der Quantisierungseffekt durch Hinzufügen von Rauschen nachgeahmt wird. Die Verfahren werden im Folgenden einzeln beschrieben.

Vortraining

Vortraining ist ein iterativer Prozess zur Bestimmung geeigneter initialer Skalierungsfaktoren. Das Skalierungsverfahren aus Abschnitt 3.4.2 ist für kleine CNNs ausgelegt. Bei tieferen Netzen verschlechtern sich die Konvergenzeigenschaften des Skalierungsverfahrens: Bspw. konvergieren bei VGG-19 nur ca. 90% aller Trainingsversuche, was auf die Initialisierung der Gewichtsfaktoren mit dem Wert 1 zurückzuführen ist. Mit steigender Anzahl an Filterkernen D pro Schicht erhöht sich die Anzahl an Summationen für jeden Punkt der Aktivierungskarte gemäß Gl. (2.4). Somit steigt die Wahrscheinlichkeit, dass ein Skalierungsfaktor mit dem Wert 1 dazu führt, dass die Werte der korrespondierenden Aktivierungskarte

weit außerhalb des gültigen Wertebereichs $[0, A_{\mathrm{q,A}} - \Delta_{\mathrm{q,A}}]$ liegen, was die Konvergenzeigenschaften stark negativ beeinflusst. Da eine Ausdehnung des Wertebereichs nicht sinnvoll ist, weil dies zu einer gröberen Quantisierung oder einem uneinheitlichen Quantisierungsschema führt, ist eine gute Initialisierung der Skalierungsfaktoren notwendig.

Beim Vortraining werden bei jeder Iteration die Skalierungsfaktoren $s_l = z_{l,10\mathcal{B}}$ für jeden Filterkern einer Schicht nach Gl. (3.5) ohne gleitende Mittelwertbildung bestimmt. Um den maximalen Skalierungsfaktor bzgl. der Aktivierungen zu bestimmen, werden zehn Chargen ausgewertet. Filterkerne, für die $Y_{l,\max} = 0$ gilt, werden zufällig reinitialisiert. Anschließend wird das CNN mit mehreren hundert Chargen trainiert. Erst dann beginnt die nächste Iteration.

Feinabstimmung

Durch das Runden zur nächsten Quantisierungsstufe bei der Quantisierung werden die globalen Effekte des Quantisierungsrauschens auf die Klassifikationsleistung nicht berücksichtigt. Diese können aufgrund eines regularisierenden Effekts positiv (s. [60]) oder negativ sein, da insbesondere Gewichte, deren Betrag kleiner als $\Delta_{\mathrm{q,W}}/2$ ist, ihren Einfluss auf die Aktivierung verlieren. Diese Gewichte sind vermehrt in den hinteren Schichten zu finden.

Daher wird in der Feinabstimmung das CNN mit einem zusätzlichen Strafterm erneut trainiert, der größere relative Fehler stärker bestraft. Der relative Fehler ist als Strafterm ungeeignet, da für Werte der Gewichte, die im Bereich des relativen Fehlers liegen, keine sinnvollen Ergebnisse zu erwarten sind, da der Gradient konstant 1 ist. Daher wird für jedes Gewicht $h \in \mathcal{H}$ des CNN der folgende Strafterm verwendet:

$$\mathcal{L}_{\mathrm{q}}(\mathcal{H}) = \sum_{h \in \mathcal{H}} -\lambda_{\mathrm{q}} \log(|h|) \cdot (h - h^{(\mathrm{q})}) + \lambda_{\mathrm{m}} \cdot \min\left\{0, |h| - A_{\mathrm{q,W}}\right\}, \tag{3.7}$$

wobei $h^{(\mathrm{q})}$ der quantisierte Wert von h ist. Der hintere Teil des Regularisierungsterms verhindert, dass die Gewichte ihren Maximalwert überschreiten. Die Straffunktion ist in Abb. 3.4 illustriert.

Die Feinabstimmung wird nach der Skalierung der Gewichte $\mathbf{k}_l \leftarrow \mathbf{k}_l s_l$ und $b_l \leftarrow b_l s_l$ und vor dem Zuordnen der Gewichte zu den Quantisierungsstufen ausgeführt. Dabei wird das CNN schichtweise mit dem Ziel

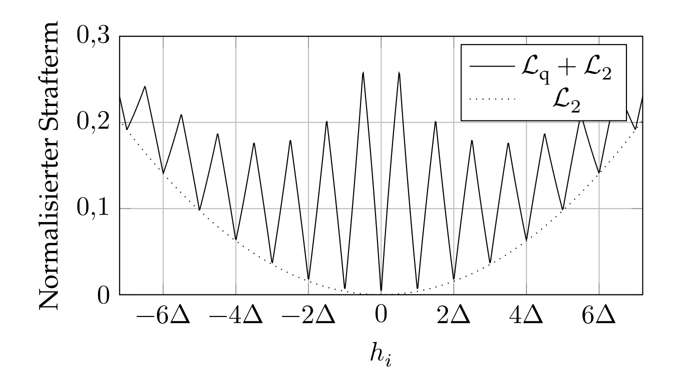


Abbildung 3.4 Strafterm der Feinabstimmung für eine 4-Bit-Quantisierung mit L_2-Regularisierung.

trainiert, den mittleren quadratischen Fehler zwischen der ursprünglichen Ausgabe und der feinabgestimmten Ausgabe zu minimieren.

Channel Pruning

Trainierte CNNs besitzen eine hohe Redundanz aufgrund ihrer Überparametrisierung, die für ein erfolgreiches Training notwendig ist [33]. Da die CNNs in dieser Arbeit für eine kritische Hardwareumgebung entworfen werden, ist die Verwendung eines Verfahrens zur Redundanzreduktion sinnvoll, um den Speicherbedarf zu senken und die Geschwindigkeit der Inferenz zu erhöhen. Das *Channel-Pruning*-Verfahren nach [64] verspricht eine Beschleunigung um den Faktor 5 bei geringen Verlusten in der Genauigkeit und wird im Folgenden beschrieben.

Das Verfahren zielt darauf ab, einzelne Filterkerne $\mathbf{k}_l^{(i)}$ aus einer Schicht i zu entfernen. Dies reduziert nicht nur die Anzahl $L^{(i)}$ der Filterkerne und die dazugehörenden Gewichte dieser Schicht, sondern auch die Tiefe $L^{(i)}$ des Aktivierungstensors $\mathbf{Y}^{(i)}$ und die Tiefe der Filterkerne $D^{(i+1)}$ der darauffolgenden Schicht (vgl. Abb. 2.3).

Beim *Channel Pruning* werden jeweils zwei Blöcke aus einem CNN extrahiert. Ein Block besteht aus einer Faltungsschicht und ggf. der Aktivierung, einer *Pooling*-Schicht und einer Chargennormalisierungsschicht. Zwischen die beiden Blöcke wird eine sog. *beta*-Schicht eingefügt. Die-

se skaliert die Aktivierungskarte $\mathbf{Y}_l^{(i)}$ des ersten Blocks mit dem Faktor $\beta_{l,\text{CP}} \geq 0$ mit $l = 1, ..., L$, wodurch sich $\mathbf{X}_l^{(i+1)} = \beta_{l,\text{CP}} \cdot \mathbf{Y}_l^{(i)}$ als Eingabe des zweiten Blocks ergibt. Die Ausgabe des zweiten Blocks $\mathbf{Y}^{(i+1)}$ wird weiterhin gemäß Gl. (3.3) berechnet.

Das so erhaltene Subnetz wird mit dem Ziel trainiert, den mittleren quadratischen Fehler der neuen Ausgabe und der Originalausgabe unter der Nebenbedingung zu minimieren, dass der Vektor $\boldsymbol{\beta}_{\text{CP}} = [\beta_{1,\text{CP}}, ..., \beta_{L,\text{CP}}]$ dünn besetzt ist. Daher wird ein L_1-Regularisierer für die Faktoren $\beta_{l,\text{CP}}$ verwendet. Nach dem Training des Subnetzes können alle Filterkerne $\mathbf{k}_l^{(i)}$, für deren Faktor $|\beta_{l,\text{CP}}| < \epsilon_{\text{CP}}$ gilt, entfernt werden, da die korrespondierende Ausgabekarte $\mathbf{Y}_l^{(i)}$ einen verschwindenden Einfluss auf die folgenden Blöcke bzw. Schichten hat. Das Entfernen führt ferner dazu, dass der Eingabetensor des zweiten Blocks eine geringere Tiefe D besitzt und die entsprechenden Karten von $\mathbf{k}_l^{(i+1)}$ entfernt werden müssen. Dies reduziert zusätzlich den Speicherbedarf für Aktivierungen und Gewichte der nächsten Faltungsschicht.

Nach dem Training ist es erforderlich, die gefundenen Faktoren $\boldsymbol{\beta}_{\text{CP}}$ mit den Gewichten des ersten Blocks zu multiplizieren. Wird Chargennormalisierung verwendet, werden die entsprechenden Parameter $\gamma_{\text{BN},l} \leftarrow \beta_{l,\text{CP}} \cdot \gamma_{\text{BN},l}$ und $\delta_{\text{BN},l} \leftarrow \beta_{l,\text{CP}} \cdot \delta_{\text{BN},l}$ skaliert (vgl. Gl. (2.5)). Liegt keine Chargennormalisierungsschicht vor wie bspw. bei konvertierten zu quantisierenden Netzen, werden die entsprechenden Gewichte mit $\mathbf{k}_l^{(i)} \leftarrow \beta_{l,\text{CP}} \cdot \mathbf{k}_l^{(i)}$ und $b_l \leftarrow \beta_{l,\text{CP}} \cdot b_l$ skaliert. Letzteres ist potentiell schädlich für die Quantisierung, da implizit die Skalierungsfaktoren bzgl. der Gewichte verkleinert werden, was zu kleineren Gewichten und den dazugehörigen Schwierigkeiten bei der Quantisierung führt.

Hinzugabe von Rauschen

Die folgende Erweiterung wurde bisher noch nicht veröffentlicht. Das vorgestellte Quantisierungsverfahren hat das Ziel, dass insbesondere der Einfluss des Clippings bei der Quantisierung und die Quantisierung kleiner Aktivierungen bereits während des Trainings berücksichtigt werden. Dabei bleibt der Einfluss der eigentlichen Quantisierung der Gewichte und Aktivierungen unberührt.

Um den Einfluss der Quantisierung nachzubilden, wird daher während des Trainings gleichverteiltes Rauschen auf die Gewichte und Akti-

vierungen addiert. Das Rauschen folgt für die Aktivierungen der Gleichverteilung $\mathtt{n}_{\mathrm{q,A}} \sim \mathcal{U}(-2^{-L_{\mathrm{q,F,A}}-1}, +2^{-L_{\mathrm{q,F,A}}-1})$ und für die Gewichte $\mathtt{n}_{\mathrm{q,W}} \sim \mathcal{U}(-2^{-L_{\mathrm{q,F,W}}-1}, +2^{-L_{\mathrm{q,F,W}}-1})$, wobei bei ersterem das Rauschen vor der Aktivierungsfunktion (s. Gl. (3.6)) hinzugefügt wird. Hierdurch wird insbesondere bei Aktivierungen, die kleiner als $\Delta_{\mathrm{q,A}}/2$ sind, das Rauschen entfernt. Das Quantisierungsrauschen für Gewichte $\mathtt{n}_{\mathrm{q,W}}$ wird mit dem korrespondierenden Skalierungsfaktor normiert (s. Gl. (3.8)), da in der Lernphase die Skalierung nach der Faltungsoperation ausgeführt wird. Es gilt für $s_l > 0$ und eine ReLU-Aktivierung

$$
\begin{aligned}
s_l \mathbf{Y}_l &= \sigma_{\mathrm{akt}}(\mathbf{X} ** (s_l \mathbf{k}_l + \mathtt{n}_{\mathrm{q,W}}) + s_l b_l + \mathtt{n}_{\mathrm{q,W}}) \\
\Leftrightarrow \mathbf{Y}_l &= \sigma_{\mathrm{akt}}\left(\mathbf{X} ** \left(\mathbf{k}_l + \frac{\mathtt{n}_{\mathrm{q,W}}}{s_l}\right) + b_l + \frac{\mathtt{n}_{\mathrm{q,W}}}{s_l}\right). \qquad (3.8)
\end{aligned}
$$

3.5 CNN Topologie

Im Folgenden wird ein heuristisches Verfahren zur Bestimmung einer CNN-Topologie beschrieben. Dieses Verfahren wurde zuerst in [191] publiziert. In [190] wird eine Erweiterung und in [193] eine Adaption auf *MobileNet*-Architekturen nach [72] vorgestellt. Die Bestimmung einer geeigneten CNN-Topologie stellt insofern eine Herausforderung dar, da einerseits eine hohe Testgenauigkeit verlangt wird und andererseits für die Implementierung auf FPGAs oder mobilen Geräten strenge Hardwarelimitationen gelten. Da die bekannten Standard-CNNs oft überdimensioniert sind, wird die Topologie hier datensatzspezifisch entworfen, um den Ressourcenbedarf zu minimieren.

Ferner wird untersucht, ob tiefere oder breitere CNNs besser für die gegebenen Klassifikationsaufgaben geeignet sind. Alom et al. zeigen in [2], dass diese Fragestellung abhängig vom Lerndatensatz ist. Außerdem zeigen sie, dass, während breitere Netze dazu neigen, bestimmte Bildeigenschaften auswendig zu lernen, tiefere Netze eine bessere Fähigkeit zum Generalisieren haben, da Merkmale auf höheren Abstraktionsebenen erlernt werden.

3.5.1 Stand der Technik

In den frühen 2000ern wurden genetische bzw. evolutionäre Heuristiken für die Anordnung von Neuronen sowie für die Bestimmung der Gewichte ihrer Verbindungen verwendet [46, 97, 150]. Da diese Verfahren allerdings nur langsam konvergieren, wurden diese obsolet, als begonnen wurde, neuronale Netze mit Fehlerrückführung (engl. *backpropagation*) zu trainieren. Neuere Verfahren verwenden dagegen Verstärkungslernen, um geeignete CNNs zu finden [180].

In [172] wird ein genetischer Algorithmus dazu verwendet, um Faltungsoperationen zwischen vordefinierten CNN-Blöcken, die in einer festen Topologie verankert sind, miteinander zu vernetzen. Dagegen verwenden Suganuma et al. in [154] einen genetischen Algorithmus, der auf einer zweidimensionalen Anordnung von Knoten aufbaut. Die Knoten enthalten typische CNN-Operationen wie Faltungs-, Pooling- oder Summationsschichten.

Ein Verfahren, das auf differentieller Evolution basiert, wird in [44] vorgestellt. Hierbei wird die Topologie eines Autoencoders für die Entfernung von Rauschen genetisch und die Gewichte trainingsbasiert bestimmt. Aufgrund der Fitnessfunktion wird der Ressourcenbedarf drastisch reduziert, da vollständig verbundene Schichten zu Strukturen ausgedünnt werden, die denen von Faltungsschichten ähneln.

Allen vorgestellten Algorithmen ist gemeinsam, dass sie versuchen, die Validierungsgenauigkeit zu maximieren, indem für jede neue Topologie das zugehörige CNN mehrere Durchläufe trainiert wird, bevor die Genauigkeit bestimmt wird. Die Evaluierung der Fitness nimmt somit einen Großteil der Rechenzeit dieser Metaheuristiken in Anspruch.

3.5.2 Heuristischer CNN-Entwurf

Das vorgestellte heuristische Verfahren basiert auf differentieller Evolution [94], da diese Methodik gewöhnlicherweise mindestens genauso gut abschneidet wie eine Partikelschwarmoptimierung oder andere genetische Evolutionsstrategien [39]. Die differentielle Evolution ist eine populationsbasierte Metaheuristik, die mittels Gradientenabstieg eine Fitnessfunktion $F_{\mathrm{MH}}(\cdot)$ mithilfe einer Menge an Kandidaten, der sog. *Po-*

pulation $\mathfrak{P}_t$ maximiert. Der Gradient $g_{\mathrm{MH},i}$ bzgl. eines Merkmalsvektors $\mathbf{t}_{\mathrm{MH},i}$ wird i. A. gemäß [133]

$$g_{\mathrm{MH}}(\mathbf{t}_{\mathrm{MH},i}) = \sum_{j \in \mathfrak{P}_t} \left(\tilde{F}_{\mathrm{MH}}(j) - \tilde{F}_{\mathrm{MH}}(i) \right) \cdot (\mathbf{t}_{\mathrm{MH},j} - \mathbf{t}_{\mathrm{MH},i}) \tag{3.9}$$

bestimmt, wobei $\tilde{F}_{\mathrm{MH}}(\cdot)$ die auf den Bereich $[0, 1]$ normierte Fitness einer Population ist. Eine Iteration der differentiellen Evolution besteht aus den drei Schritten Vektoraktualisierung (engl. *vector updating*), Vektorsprung (engl. *vector jumping*) und Erneuerung (engl. *refreshing*). Während Gl. (3.9) bei der Vektoraktualisierung angewandt wird, dienen Vektorsprung und Erneuerung zur Vermeidung lokaler Extrema durch zufällige Modifikationen (Vektorsprung) oder Änderung des Evaluationspunktes (Erneuerung).

In dieser Arbeit wird ein zweistufiger Ansatz für die differentielle Evolution verwendet, um die Anzahl der Schichten $L_{\mathrm{MH},i}$ und der Filterkerne $\mathbf{b}_{\mathrm{MH},i}$ pro Schicht als Merkmale getrennt voneinander zu optimieren. Während für die Schätzung des Gradienten der Tiefe $L_{\mathrm{MH},i}$ die gesamte aktuelle Population $\mathfrak{P}_t$ verwendet wird, werden für die Breite nur die Kandidaten mit der gleichen Tiefe verwendet.

Der Algorithmus besteht nach der Initialisierung aus Iterationen, in denen zuerst ein Gradientenschritt bzgl. der Tiefe und dann bzgl. der Breite berechnet wird und anschließend ein Teil der Population $\mathfrak{P}$ zufällig variiert wird. Die einzelnen Schritte werden im Folgenden für gemäß Abschnitt 3.4 quantisierte CNNs näher beschrieben.

Aufbau der CNNs und Initialisierung

Als Tiefe bzw. Länge des i-ten CNN L_{MH_i} wird im Folgenden die Anzahl an Blöcken vor dem Ausgabeblock verstanden, welcher aus einer (1×1)-Faltungsschicht, einer GAP-Schicht und einer geeigneten Aktivierungsfunktion besteht. Ein Block besteht ansonsten aus einer modifizierten Faltungsschicht (vgl. Gl. (3.3)) und ggf. einer *Max-Pooling*-Schicht. Jeder dieser Blöcke hat die Anzahl an Filterkernen als Parameter. Um zu große und kleine CNNs zu vermeiden, wird die Länge $L_{\mathrm{MH},i}$ auf 3 bis 14 Blöcke limitiert. Die Anzahl der Filterkerne wird durch den Vektor $\mathbf{b}_{\mathrm{MH},i}$ mit den $L_{\mathrm{MH},i}$ Einträgen $b_{\mathrm{MH},i,l} \in [32; 512]$ mit $l = 0, ..., L_{\mathrm{MH},i}$ parametrisiert, wobei jeder Eintrag der Anzahl der Filterkerne der korrespondierenden Schicht entspricht. Die Beschränkungen bzgl. der Breite

der Schichten ergeben sich aus den Ergebnissen in Abschnitt 4.4.1. *Max-Pooling*-Schichten treten nur in den Blöcken auf, die mit den Stellen der *Max-Pooling*-Schichten eines VGG-19-Netzes korrespondieren. Um die Dreidimensionalität der Aktivierungstensoren nicht zu zerstören, werden keine *Max-Pooling*-Schichten mehr verwendet, wenn die Höhe H bzw. Breite B des Tensors den Wert 4 unterschreiten würde. Eine Topologie ist somit durch die beiden Parameter $L_{\mathrm{MH},i}$ und $\mathbf{b}_{\mathrm{MH},i}$ darstellbar.

Bei der Initialisierung werden $N_{\mathrm{MH},0}/2$ Topologien zufällig mit zulässigen Parametern erzeugt und für $n_{\mathrm{MH},0}$ Durchläufe trainiert, bevor die Fitness bestimmt wird. Anschließend werden die Topologien dupliziert, um $N_{\mathrm{MH},0}$ Initialtopologien zu erhalten. Die Menge an Topologien wird als Population $\mathfrak{P}_0$ bezeichnet.

Fitnessfunktion $F_{\mathrm{MH}}(\cdot)$

Die Fitnessfunktion aggregiert die Validierungsergebnisse und den Ressourcenbedarf, um eine CNN-Topologie i zu bewerten. Hierzu wird die an einem Validierungsdatensatz ermittelte Genauigkeit $a_{\mathrm{MH},i}$ und die Kreuzentropie $\mathrm{H}_{\mathrm{MH},i}$ ermittelt. Der Ressourcenbedarf wird anhand des Speicherbedarfs $s_{\mathrm{MH},i}$ für Aktivierungen und Gewichte in KB[1] sowie der Gesamtzahl an MAC-Operationen $o_{\mathrm{MH},i}$ für ein Bild in der Inferenz bestimmt (vgl. Abschnitt 2.2.2). Die Fitnessfunktion ist gegeben als

$$F_{\mathrm{MH}}(i) = \frac{\lambda_\mathrm{a}}{1 - a_{\mathrm{MH},i} + \epsilon} + \frac{\lambda_\mathrm{H}}{\mathrm{H}_{\mathrm{MH},i} + \epsilon} - \lambda_\mathrm{s} \log_{10}(s_{\mathrm{MH},i}) - \lambda_\mathrm{o} \log_{10}(o_{\mathrm{MH},i}), \tag{3.10}$$

wobei $\lambda_{\{\mathrm{a},\mathrm{H},\mathrm{s},\mathrm{o}\}}$ Gewichtungsparameter sind und λ_s sowie λ_o um den Faktor 10 erhöht werden, wenn der maximale Speicher $s_{\mathrm{MH}}^{\mathrm{max}}$ überschritten wird. Die einzelnen Terme der Fitnessfunktion verhindern, dass die Fitness bei sinkender Genauigkeit durch die Verkleinerung der Topologie verbessert wird.

Iteration

Eine Iteration besteht aus drei Schritten und bildet eine Population $\mathfrak{P}_t$ auf

[1] Bei der Evaluation wird eine 8-Bit-Quantisierung verwendet. Damit entspricht die Anzahl der Gewichte und Aktivierungen ihrem Speicherbedarf in Bytes.

eine Population $\mathfrak{P}_{t+1}$ ab. Hierfür wird die Population in Teilpopulationen $\mathfrak{P}_t^{(l)}$ unterteilt, deren Topologien die gleiche Tiefe $L_{\mathrm{MH}} = l$ haben.

Im ersten Schritt wird für jede Teilpopulation $\mathfrak{P}_t^{(l)}$ eine Topologie i mittels Roulette-Wheel-Selektion [69] gemäß der Wahrscheinlichkeitsverteilung

$$P(i) = \frac{\exp\{F_{\mathrm{MH}}(i)\}}{\sum_{j\in\mathfrak{P}_t^{(l)}} \exp\{F_{\mathrm{MH}}(j)\}} \text{ mit } i \in \mathfrak{P}_t^{(l)} \tag{3.11}$$

ausgewählt. Anschließend wird ein Gradient $g_{\mathrm{MH}}(\mathbf{L}_{\mathrm{MH},i})$ gemäß Gl. (3.9) gebildet. Dieser wird mit $|\mathfrak{P}_t^{(l)}|^{-1}$ normiert, ggf. bei -2 bzw. $+2$ abgeschnitten und stochastisch gerundet. Je nach Wert des Gradienten werden nun der i-ten Topologie am Ende Blöcke mit Faltungsschichten hinzugefügt oder entfernt.

Im zweiten Schritt wird auf die gleiche Weise ein Gradient bzgl. der Breite $\mathbf{b}_{\mathrm{MH},i}$ innerhalb jeder Teilpopulation $\mathfrak{P}_t^{(l)}$ berechnet. Hierfür werden ebenfalls Topologien gemäß Gl. (3.11) ausgewählt. Der Gradient wird mit $25/|\mathfrak{P}_t^{(l)}|$ skaliert und der Betrag der Einträge des Gradienten wird auf 64 beschränkt. Entsprechend dem Vorzeichen und den Werten des skalierten Gradienten werden den Schichten der ausgewählten Topologie Filterkerne $\mathbf{k}_{l,d}$ hinzugefügt oder daraus entfernt. Dies erfordert das entsprechende Verändern der Tiefe der Filterkerne der darauffolgenden Schicht (vgl. Abb. 2.3).

Der dritte Schritt ist eine zufällige Mutation der Breite und Tiefe von insg. zwei gemäß Roulette-Wheel-Selektion ausgewählten Topologien, um die Diversität der Gesamtpopulation zu erhöhen. Hierbei werden den Topologien mit jeweils einer Wahrscheinlichkeit von 1/3 eine Schicht hinzugefügt, eine Schicht entfernt oder es wird gar keine Operation ausgeführt. Darüber hinaus wird die Breite jeder Schicht $\mathbf{b}_{\mathrm{MH},i}$ um einen zufälligen Wert zwischen -25 und $+25$ geändert. Die jeweiligen Zahlenwerte wurden empirisch ermittelt.

Die ersten beiden Schritte sind als Vektoraktualisierung und der dritte Schritt ist als Vektorsprung interpretierbar. Eine Vektorerneuerung wird nicht durchgeführt. Nach jedem Iterationsschritt wird die Menge der Individuen der Gesamtpopulation $\mathfrak{P}_t$ um die Schlechtesten reduziert, wodurch die Konvergenzgeschwindigkeit erhöht wird.

Training

Nach jeder Operation einer Iteration wird das aus der Topologie resultierende CNN für $n_{\mathrm{MH},t}$ Durchläufe trainiert. Dabei wird die Anzahl der Durchläufe linear bei jeder Iteration erhöht. Das Training beginnt jeweils mit einer konstanten Initiallernrate, die kosinusartig abfällt.

Besonders an diesem Ansatz ist, dass die Gewichte nach jeder Veränderung aus dem Elternetz geladen werden und das resultierende CNN kurz damit trainiert wird, bevor die Fitness gemessen wird. Dadurch reduziert sich die Dauer für die Auswertung der Fitnessfunktion für eine Topologie deutlich. Bei einer Verkleinerung der Breite werden die Filterkerne mit der geringsten Vektornorm $||\cdot||_2$ entfernt und bei einer Vergrößerung werden zufällig initialisierte Kerne hinzugefügt. Dies ist zusammen mit dem Lernratenabfall als Initialisierung der Lernrate interpretierbar und folgt den Beobachtungen von [101].

Nach den T Iterationen findet ein abschließendes Training der fittesten Topologie mit $n_{\mathrm{MH},T+1}$ Durchläufen statt.

3.6 Verarbeitungskette

Für das Lösen einer Bildklassifikationsaufgabe mit quantisierten CNNs mittels der in diesem Kapitel vorgestellten Verfahren ergibt sich eine Verarbeitungskette, die im Folgenden beschrieben und in Abb. 3.5 dargestellt wird. Die Kette ist in die drei thematischen Blöcke *Datensatzanalyse*, *Training* und *Nachverarbeitung* gegliedert.

Für die Datensatzanalyse wird eine Vorverarbeitungsstrategie mit aSVST bestimmt. Anschließend wird für den vorverarbeiteten Datensatz eine Augmentierungsstrategie mittels FAARO ermittelt.

Im Trainingsblock erfolgt entweder das Vor- und Haupttraining des gemäß Abschnitt 3.4.2 modifizierten CNN oder das trainierte quantisierte CNN wird über die Metaheuristik (s. Abschnitt 3.5.2) ermittelt. Letzteres beinhaltet ebenfalls im Initialisierungsschritt ein Vortraining.

Im letzten Schritt erfolgt die Umwandlung auf ein 8-Bit CNN, indem zunächst die Skalierungsfaktoren mit den korrespondierenden Gewichten multipliziert werden, um konventionelle Faltungsschichten zu erhalten (Konvertierung). Anschließend wird das CNN optional mit *Channel Pru-*

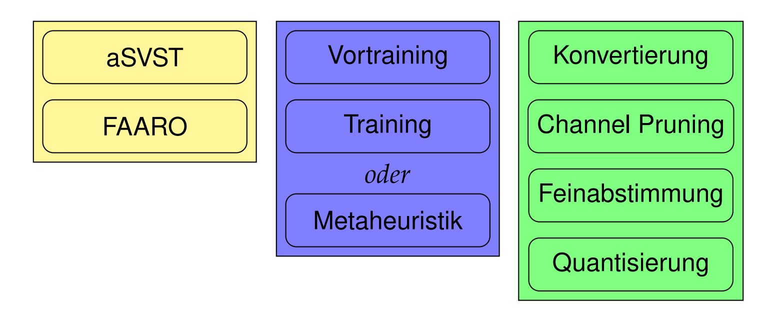


Abbildung 3.5 Kategorische Unterteilung der Verarbeitungskette in datensatzspezifische Verarbeitung der Bilddaten (gelb), Training mit dem Quantisierungsverfahren (blau) und Nachverarbeitung des CNN (grün).

ning verkleinert und die Gewichte werden feinabgestimmt. Zum Schluss werden die Gewichte auf die Quantisierungsstufen abgebildet.

4 Evaluation der Methoden für effiziente CNNs

In diesem Kapitel werden die Verfahren aus Kapitel 3 evaluiert. Die allgemeine Simulationsumgebung und die verwendeten Datensätze werden in Abschnitt 4.1 beschrieben. Je nach Verfahren werden spezifischere Beschreibungen des Versuchsaufbaus in den jeweiligen Unterkapiteln gegeben. Anschließend wird in Abschnitt 4.2 der Einfluss der Chargengröße auf das Training mit Augmentierung analysiert. In Abschnitt 4.3 werden die Verfahren, die das Training und den Datensatz betreffen, untersucht. Neben dem Einfluss der Chargengröße bei Augmentierung wird auch eine Vorverarbeitung und Augmentierung untersucht. Bei der Vorverarbeitung wird nach einer Kombination von Bildverbesserungsverfahren gesucht. Hierfür werden einzelne Bildverbesserungsmethoden und Kombinationen davon bei der Verwendung einer Basisaugmentierung untersucht. Anschließend wird das erweiterte *faster-AutoAugment*-Verfahren untersucht. Hierbei wird nicht nur die erzielte Genauigkeit, sondern auch der Einfluss der Vorverarbeitung auf die gefundene Augmentierungsstrategie evaluiert.

In Abschnitt 4.4 wird das Quantisierungsverfahren aus Abschnitt 3.4 mit seinen Erweiterungen analysiert. Neben den erzielten Testgenauigkeiten bei einer 8-Bit-Quantisierung werden auch kleinere Bitlängen betrachtet sowie deren Einfluss auf die Zielfunktion. Abschließend wird in Abschnitt 4.5 das heuristische Verfahren zur Ermittlung der datensatzspezifischen CNN-Topologie aus Abschnitt 3.5 in Kombination mit den zuvor evaluierten Verfahren untersucht (vgl. Abschnitt 3.6).

4.1 Evaluationsumgebung

Die in Kapitel 3 beschriebenen Methoden werden mithilfe der `tensorflow 2.0`-Bibliothek in `python` implementiert. Für das Training wird eine `GeForce RTX 2080 TI` GPU verwendet. Falls nicht anders erwähnt, werden VGG-Netze [145] verwendet, wobei nach der letzten Faltungsschicht statt der drei vollständig verbundenen Schichten eine (1×1)-Faltungsschicht und eine GAP-Schicht verwendet werden, was die Anzahl der Parameter des CNN erheblich reduziert. Im Folgenden wird weiter die Bezeichnung nach Simonyan et al. [145] verwendet, auch wenn ein VGG-16-Netzwerk in dieser Arbeit nur 14 Schichten mit trainierbaren Parametern hat. Ferner wird eine L_2-Regularisierung mit dem L_2-Gewicht 10^{-5} verwendet und ein kosinusartiger Abfall der Lernrate, wobei die Initiallernrate $0{,}002$ beträgt und ein ADAM-Optimierer verwendet wird. Bei den quantisierten CNNs wird zusätzlich eine L_1-Regularisierung mit dem L_1-Gewicht 10^{-7} verwendet. Während durch die L_2-Regularisierung Ausreißer bei den Gewichten vermieden werden, sorgt der L_1-Regularisierer dafür, dass möglichst viele Gewichte dem Wert null zugeordnet werden, was unerwartete Effekte bei der Quantisierung von Gewichten $|h_i| \leq {}^{\Delta_\mathrm{w}}/_2$ verringert.

In diesem Kapitel werden vier Datensätze verwendet, die im Folgenden beschrieben werden. Die Bilder der Datensätze liegen als `uint8` vor. Um die Bilder als Eingabe für ein CNN zu verwenden, werden die Pixelwerte als Gleitkommazahl interpretiert, um den Wert 128 vermindert und mit ${}^1/_{64}$ skaliert, sodass die Pixelwerte betragsmäßig auf den Wert 2 begrenzt sind. Bei den Versuchen mit Quantisierung (s. Abschnitt 4.4) ist der Skalierungsfaktor ${}^{A_{\mathrm{q},A}}/_{256}$.

Der CIFAR-10-Datensatz [91] besteht aus 60.000 RGB-Bildern der Größe 32×32 Pixel und stellt eine Objektklassifikationsaufgabe dar, die als Benchmark für CNN-Methoden gilt. Der Datensatz ist in 50.000 Lern- und 10.000 Testbilder aufgeteilt.

Der zweite Datensatz ist der SVHN-Datensatz (*street view house numbers*) [120], der RGB-Bilder von Hausnummern von `Google Street View` enthält und bei dem Ziffern klassifiziert werden. Die Größe der Bilder beträgt ebenfalls 32×32 Pixel. Der Testdatensatz hat eine Größe von 26.032 Bilder. Um die gleiche Anzahl an Bildern pro Klasse zu erhalten, wird der Lerndatensatz um Bilder aus dem *Extra*-Datensatz vergrößert.

Insgesamt stehen 138.610 Trainingsbilder zur Verfügung. Nachteilig an diesem Datensatz ist, dass teilweise mehrere verschiedene Ziffern in einem Bild sichtbar sind und dass Ziffern in den Testdaten teilweise nicht zentriert sind. Nach [30] stellt der SVHN-Datensatz hinsichtlich der optimalen Augmentierungsstrategie einen zu CIFAR-10 orthogonalen Datensatz dar.

Da industrielle Bildverarbeitungsaufgaben mitunter deutlich weniger komplex sind als die zuvor beschriebenen Datensätze, werden im Folgenden zwei typische industrielle Datensätze beschrieben, bei denen Defekte auf metallischen Oberflächen klassifiziert werden. Der RSDDs-Datensatz (*rail surface discrete defects*) [47] enthält Aufnahmen von defektbehafteten Eisenbahnschienen. Die Bilder haben eine Auflösung von 55×1.250 Pixeln und besitzen eine Maske der gleichen Größe, die die defektbehafteten Bereiche im Bild markiert. Die Grundwahrheit erfasst jedoch nur relativ große Fehler. Bilder mit kleinen Defekten werden daher als *defektfrei* annotiert. Dies ist auf das mehrstufige Annotierungsverfahren zurückzuführen, das versucht, Rauschen zu unterdrücken. Für den Zweck der Defekterkennung werden Quadrate der Größe 55×55 Pixel aus den Bildern ausgeschnitten und anhand der Grundwahrheit in die Klassen *defektfrei* und *defektbehaftet* eingeteilt. Dies ergibt einen Lerndatensatz von 2.342 und einen Testdatensatz von 865 Bildern. Für die Eingabe in das CNN werden die Bilder auf 64×64 Pixel hochgetastet.

Der Stahloberflächendefektdatensatz der *North Eastern University* (NEU) [147] zeigt sechs verschiedene Oberflächendefekte auf jeweils 300 Grauwertbildern. Für die Eingabe in das CNN werden die Bilder auf 64×64 Pixel heruntergetastet und als RGB-Bilder interpretiert. Es werden 1.260 Bilder zum Lernen und 540 Bilder zum Testen verwendet.

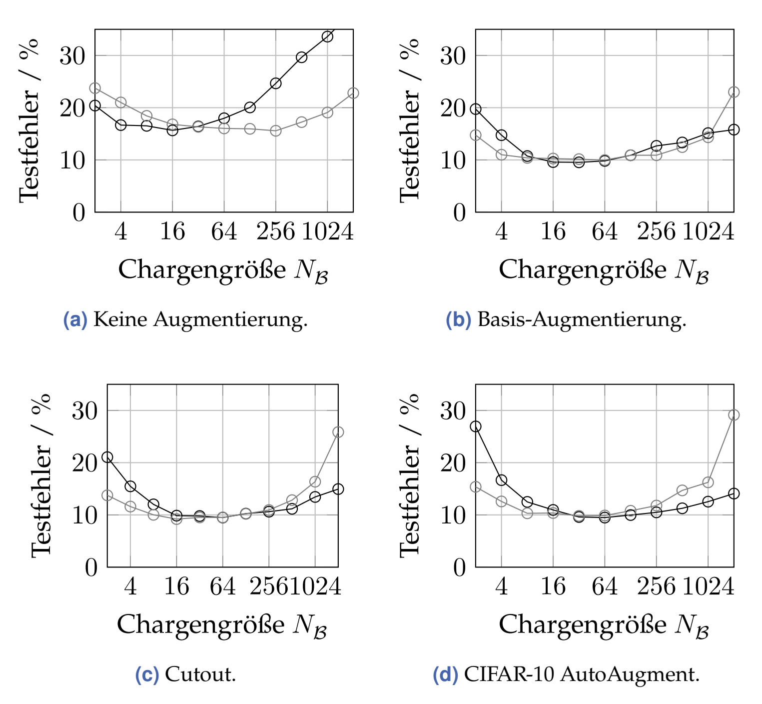
Beim RSDDs-Datensatz wird als Aktivierung der letzten Schicht eine *Sigmoid*-Funktion und als Zielfunktion die binäre Kreuzentropie verwendet. Bei allen anderen Datensätzen wird eine *softmax*-Funktion und die kategorische Kreuzentropie verwendet.

4.2 Einfluss der Chargengröße bei der Augmentierung

In diesem Abschnitt wird der Einfluss der Chargengröße und des Grades der Augmentierung auf die Klassifikationsleistung eines CNN untersucht. Hierzu werden die Überlegungen aus Abschnitt 3.1 bzgl. des Trainings verwendet. Bei den Experimenten wird ein konvolutionäres neuronales Netz mit sieben Faltungsschichten und einer GAP-Schicht mithilfe des CIFAR-10-Datensatzes [91] untersucht. Die vier betrachteten Augmentierungsgrade sind *keine Augmentierung*, Basisaugmentierung (Spiegelung und Translation), *Cutout* nach [35] mit Basisaugmentierung und die *AutoAugment*-Pipeline für CIFAR-10 nach [30]. Die Lernphase besteht aus 100 Durchläufen. Für das Training mit fester Iterationszahl wird die Anzahl der Iterationen für eine Chargengröße von 32 berechnet, was bei CIFAR-10 1.563 Iterationen pro Durchlauf entspricht.

Voruntersuchungen haben gezeigt, dass die optimale Chargengröße $N_{\mathcal{B}}$, d. h. die Chargengröße mit niedrigstem Testfehler, stark von der gewählten Augmentierung und ggf. der Skalierung des Gradientenschritts abhängt (s. Abb. 4.1). Dies führt zu Inkonsistenzen beim Vergleich verschiedener Verfahren. Daher wird statt der konventionellen Definition eines Durchlaufs (s. Abschnitt 2.2.3) die Anzahl an Iterationen pro Durchlauf konstant gehalten. Dies führt zu den Ergebnissen aus Abb. 4.2, die darauf schließen lassen, dass der Testfehler für größere Chargengrößen bei einem höheren Augmentierungsgrad sinkt, falls die Anzahl an Iterationen pro Durchlauf konstant ist. Umgekehrt führt dies dazu, dass sich bei einer stärkeren Augmentierung die optimale Chargengröße zu höheren Werten hin verschiebt.

Sofern durch zu schwache Augmentierung keine Überanpassung auftritt, konvergiert das CNN während des Trainings schneller, was auf die weniger stark verrauschten Iterationsschritte bei großen Chargengrößen zurückzuführen ist. Jedoch ist die Chargengröße einerseits durch die Trainingshardware und andererseits durch die Dauer des Trainings beschränkt, da die Augmentierung zum Flaschenhals bzgl. der Trainingsgeschwindigkeit wird, falls eine Online-Augmentierung verwendet wird. Die Rechenzeit nimmt ab einer Chargengröße von 32 exponentiell zu: Bei der verwendeten Hardware entspricht eine Verachtfachung der Chargen-



(a) Keine Augmentierung.

(b) Basis-Augmentierung.

(c) Cutout.

(d) CIFAR-10 AutoAugment.

Abbildung 4.1 Testfehler für CIFAR-10 über die Chargengröße $N_\mathcal{B}$ bei verschiedenen Augmentierungsgraden mit linearer (grau) und ohne (schwarz) Gradientenskalierung nach dem konventionellen Trainingsschema, d. h. mit variabler Anzahl an Iterationen pro Durchlauf (vgl. Abschnitt 2.2.3). Je nach Parameterwahl liegt die optimale Chargengröße zwischen 16 und 256.

größe in etwa einer Verzehnfachung der Rechendauer. Dieser Effekt tritt bei einer Offline-Augmentierung nicht in dieser Form auf, jedoch legen weitere Versuche nahe, dass der Datensatz durch Augmentierung um den Faktor 64 vergrößert werden muss, damit die Offline-Augmentierung denselben Effekt wie eine Online-Augmentierung aufweist, was bei der Größe der vorliegenden Datensätze nicht sinnvoll ist. Ferner erhöht sich die Rechendauer durch den größeren Offline-Datensatz. Nähere Informationen zu dieser Thematik sind im Anhang A.1 aufgeführt. Die Ergebnisse deuten darauf hin, dass Online-Augmentierung nicht nur die Testgenauigkeit nach dem Training erhöht, sondern auch das Training beschleunigt.

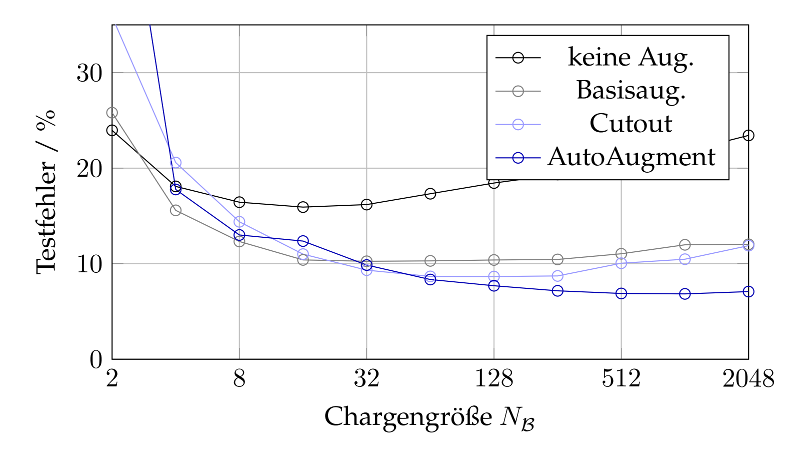
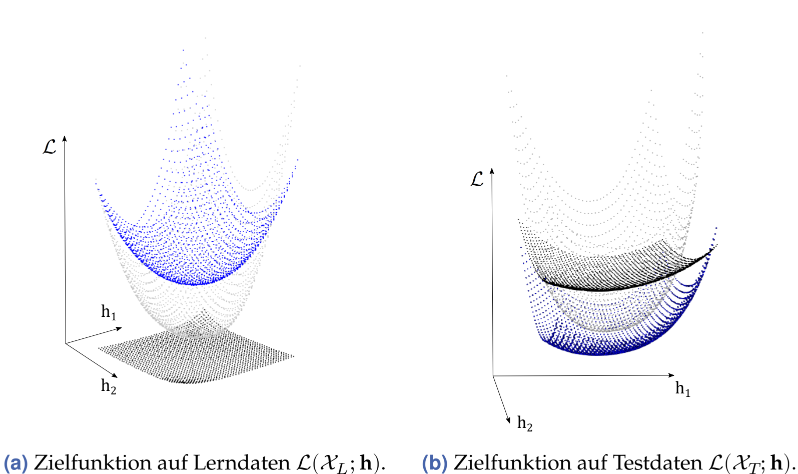


Abbildung 4.2 Testfehler für CIFAR-10 [91] über die Chargengröße $N_\mathcal{B}$ bei verschiedenen Augmentierungsgraden mit konstanter Anzahl an Iterationen pro Durchlauf.

Augmentierung hat einen Einfluss auf die Zielfunktion während des Trainingsvorgangs und auf das gefundene Optimum nach dem Training, nicht nur, weil die Zielfunktion von den Eingangsdaten $\mathcal{X}$ abhängt, sondern auch, weil die augmentierten Bilddaten die Gewichte $\mathcal{H}$ des CNN beeinflussen. In Abb. 4.3 ist die Zielfunktion $\mathcal{L}$ des neuronalen Netzes im Optimum für den Lern- und Testdatensatz nach dem Verfahren aus [100] visualisiert. Das Training wurde mit konstanter Anzahl an Iterationen mit einer Chargengröße von $N_\mathcal{B} = 128$ durchgeführt. Je stärker die Augmentierung, desto schwieriger wird die Lernaufgabe, was in Abb. 4.3a zur Folge hat, dass die Zielfunktionen bei stärkerer Augmentierung zu höheren Werten verschoben wird. Die Zielfunktion bzgl. des Lerndatensatzes hat die geringsten Werte und gleicht einer Ebene, was ein Zeichen für Überanpassung ist und ein Hindernis bei der Optimierung darstellt, da kein ausgeprägtes Minimum vorhanden ist. Die Reihenfolge der Zielfunktionen bei Testdaten ist umgekehrt. Dies weist auf eine geringere Generalisierungslücke Γ bei einer besser geeigneten Augmentierung hin.

Insgesamt ist die Verwendung einer festen Anzahl an Iterationen vorteilhaft, da sich die optimale Chargengröße $N_\mathcal{B}$ aus der Hardwarelimitierung ergibt und bei hinreichend großem Augmentierungsgrad ab dem Wert 128 einen verschwindend geringen Einfluss hat. Daher wird



(a) Zielfunktion auf Lerndaten $\mathcal{L}(\mathcal{X}_L; \mathbf{h})$. **(b)** Zielfunktion auf Testdaten $\mathcal{L}(\mathcal{X}_T; \mathbf{h})$.

Abbildung 4.3 Zielfunktion für verschiedene Augmentierungsgrade (schwarz: keine Augmentierung, grau: Basisaugmentierung, blau: AutoAugment) für CIFAR-10.

im Folgenden die Chargengröße zu 128 gewählt. Ferner wird Online-Augmentierung, d. h. Bilddatenaugmentierung während des Trainings, verwendet.

4.3 Vorverarbeitungs- und Augmentierungsverfahren

Im Folgenden werden die Methoden für die Konstruktion invarianter Merkmale untersucht. In Abschnitt 4.3.1 wird das Vorverarbeitungsverfahren aus Abschnitt 3.2.2 hinsichtlich der Qualität der Bilder und der resultierenden Testgenauigkeit untersucht. Die Suche nach der optimalen Augmentierungsstrategie $\mathcal{T}_{\text{aug}}$ sowie der Einfluss von Bildverbesserung als Vorverarbeitungsschritt darauf wird anschließend in Abschnitt 4.3.2 thematisiert.

4.3.1 Vorverarbeitung

Neben den Versuchen aus [188], die die Maximalparameterbestimmung, Voruntersuchungen, die Analyse der Bilder und der erzielten Genauigkeit untersuchen, werden in diesem Abschnitt auch Approximationen des Vorverarbeitungsverfahrens und das aSVST-Verfahren betrachtet. Dagegen wird die in [188] betrachtete Fensterung in dieser Arbeit nicht berücksichtigt, da die Ergebnisse keine wesentlichen Verbesserungen mit sich bringen und schwer interpretierbar sind.

Sowohl für das Training zur Bestimmung der Vorverarbeitungsgewichte γ_V und $\boldsymbol{\theta}_\mathrm{V}$ als auch für das finale Training wird ein VGG-16-Netz verwendet. Ersteres wurde auf *ImageNet* [32] vortrainiert. Beide Lernphasen sind 50 Durchläufe lang. Es wird eine Basisaugmentierung verwendet, die aus Spiegelung (außer bei SVHN), einem zufälligen Ausschneiden eines Bildausschnittes und *Cutout* (nicht bei RSDDs und NEU) besteht. In diesem Abschnitt wird zunächst nach geeigneten Parameter der einzelnen Bildverbesserungsverfahren gesucht. Anschließend werden in Vorversuchen die Verfahren einzeln untersucht. Danach werden die durch das aSV-Verfahren verarbeiteten Bilder analysiert und dann wird die Auswirkung der Verfahren auf die Testgenauigkeit untersucht. Im letzten Teil wird das aSVST-Verfahren betrachtet.

Parameter der Bildverbesserungsverfahren

Die für die Vorverarbeitung verwendeten Bildverbesserungsverfahren unterscheiden sich mitunter stark bezüglich ihrer Parameter. Im Folgenden werden Versuche durchgeführt, um die Maximalparameter $\mathbf{a}_{\mathrm{V},i}^{\max}$ aus Gl. (3.2) für jedes Verfahren mithilfe des CIFAR-10-Datensatzes zu bestimmen. Dies geschieht unter der Annahme, dass gut geeignete Parameter die erzielte Genauigkeit am stärksten verbessern bzw. dass das zu $\mathbf{a}_{\mathrm{V},i}^{\max}$ gehörende Verfahren i einen höheren Anteil $\theta_{\mathrm{V},i}$ besitzt. Letzteres rührt daher, dass der positive Einfluss eines Verfahrens genau dann am größten ist. Für die parameterfreien Verfahren der Histogrammspreizung und der Histogrammegalisierung sind keine Parameter zu ermitteln. Soweit möglich, werden Standardparameter verwendet.

Ein Großteil der verwendeten Verfahren hat die Größe eines Filterkerns und die Standardabweichung einer Gaußfunktion σ_Gauss als Parameter. Da diese mit der Bildgröße korrespondieren, werden für alle Verfahren

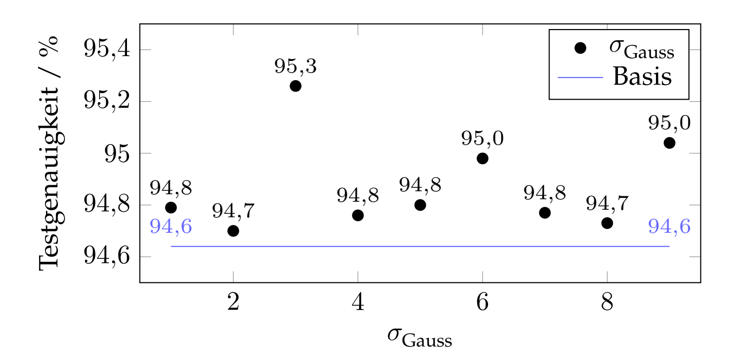


Abbildung 4.4 Testgenauigkeit über der Standardabweichung σ_{Gauss} des Gaußfilters mit AutoAugment nach [30]. Für die Basisgenauigkeit wird keine Vorverarbeitung verwendet. Für das Training wurde ein WideResNet verwendet.

außer der Homogenisierung die gleichen Werte verwendet, deren Ermittlung im Folgenden beschrieben wird. Hierfür wird das Verfahren für ungerade Filterkerngrößen von 3 bis 15 angewandt, um die besten Filterkerne zu bestimmen. Für die verwendeten Datensätze mit Bildern erweisen sich Kerne der Größe 3×3 als optimal. Für die Homogenisierung wird die Filtergröße 21×21 verwendet. Die Standardabweichung der Bildverbesserungsverfahren, die ein Gaußfilter beinhalten, wurde für Ganzzahlen von 1 bis 9 getestet (s. Abb. 4.4). Die optimale Standardabweichung ist 3 und für die Homogenisierung beträgt diese 9. Bei diesem Versuch ändert sich die Verteilung der Gewichte $\theta_{\text{V},i}$ kaum.

Für die Bildverschärfung und das homomorphe Filter ist ein dritter Parameter α_{BS} bzw. α_{HF} zu bestimmen, der einen erheblichen Einfluss auf die Intensität des Filters hat. Das Ergebnis ist in Abb. 4.5 illustriert. Das Maximum des Anteils korrespondiert hier jeweils mit einem lokalen Maximum der Genauigkeit. Somit ergibt sich für die homomorphe Filterung ein geeigneter Wert für den Parameter von $0{,}2$ und für die Bildverschärfung von $0{,}6$.

Vorversuche

Zunächst werden die verwendeten Bildverbesserungsverfahren einzeln an den Datensätzen aus Abschnitt 4.1 getestet. Hierzu werden die Datensätze mit den jeweiligen Bildverbesserungsverfahren vorverarbeitet

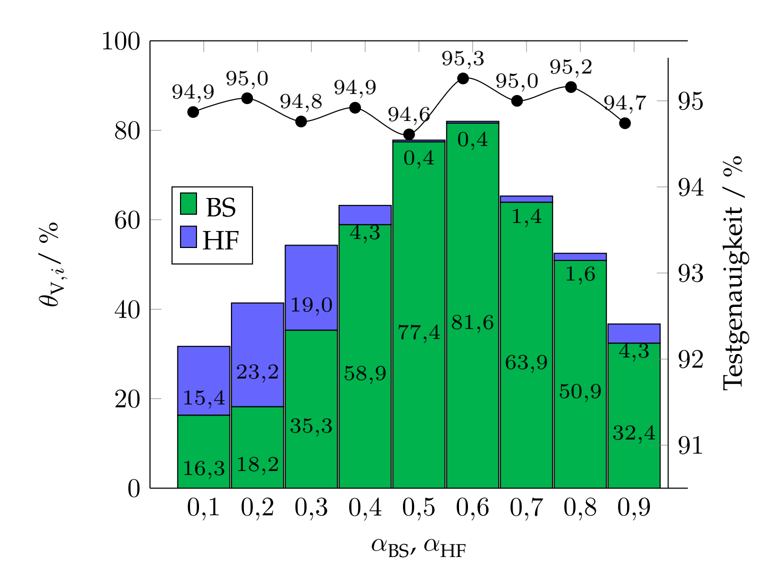


Abbildung 4.5 Testgenauigkeit (schwarze Linie) und Anteil von Bildverschärfung (grün) und homomorpher Filterung (blau) an der Vorverarbeitung bei verschiedenen Werten von α_{BS} bzw. α_{HF}. Für das Training wurde ein WideResNet verwendet.

und anschließend 50 Durchläufe mit VGG-16 trainiert. Hierbei wird eine Basisaugmentierung aus Spiegelung (außer bei SVHN), einem zufälligen Ausschneiden eines Bildausschnittes und Cutout (nicht bei RSDDs und NEU) verwendet. Die Ergebnisse sind in Tabelle 4.1 dargestellt.

Die Ergebnisse zeigen, dass der Klassifikator sehr sensibel gegenüber ungeeigneten Bildverbesserungsverfahren ist, d. h. eine schlechte Wahl kann zu dramatischen Verschlechterungen der Genauigkeiten führen, während eine gute Wahl nur geringe Verbesserungen mit sich bringt. So führt nur die homomorphe Filterung bei CIFAR-10 zu einer Verbesserung der Klassifikationsgenauigkeit. Verfahren mit einem Tiefpasscharakter (RF, MF, GF) führen zu einer Verschlechterung von 2 bis 4 Prozentpunkten. Die größte Verschlechterung ergibt sich für die Homogenisierung erster und zweiter Ordnung. Ein ähnliches Verhalten ist für den NEU-Datensatz feststellbar. Hier bringt nur die Bildverschärfung

Tabelle 4.1 Veränderung der Testgenauigkeit in Prozentpunkten der einzelnen Bildverbesserungsverfahren gegenüber dem Originaldatensatz. Verfahren, die die Genauigkeit verbessern, sind fett gedruckt.

	CIFAR-10	SVHN	NEU	RSDDs
HE	−1,70	−0,07	−0,72	**+0,39**
HS	−0,01	**+0,26**	−0,17	−0,25
CL	−2,12	−0,58	−0,43	−0,81
HF	**+0,03**	**+0,08**	−0,04	**+0,04**
BS	−0,31	**+0,15**	**+0,14**	−0,32
H1	−5,26	−0,41	−1,09	**+0,01**
H2	−60,23	−11,65	−8,41	−20,18
RF	−2,72	**+0,17**	−0,70	−0,19
MF	−3,64	**+0,12**	−0,65	−0,20
GF	−2,83	**+0,15**	−0,57	−0,27
MH	−2,90	−1,04	−4,38	−0,58
BF	−1,39	**+0,11**	−0,56	−0,30

eine Verbesserung. Bis auf H1, H2 und MH liegt die Verschlechterung der anderen Verfahren aber unter einem Prozentpunkt.

Beim SVHN-Datensatz führt eine Vielzahl unterschiedlicher Vorverarbeitungsverfahren zu einer Verbesserung. Auch hier führen die Homogenisierungen zu einer Verschlechterung. Dagegen hat beim RSDDs-Datensatz lediglich die Histogrammegalisierung einen merklich positiven Einfluss auf das Ergebnis. Während die Homogenisierung zweiter Ordnung zu einem hohen Genauigkeitsverlust führt, hat die Homogenisierung erster Ordnung quasi keinen Einfluss auf die Genauigkeit.

Obwohl der MH-Operator nur ein Kantenbild ausgibt, ist die Verschlechterung durch dieses Verfahren bis auf beim NEU-Datensatz überraschend gering, weil Schriftzeichen bzw. Ziffern (SVHN), Objekte (CIFAR-10) und Defekte (RSDDs) noch durch ihre Kontur erkannt werden können. Dies ist aber bei Texturen (NEU) schwieriger.

Dieser Vorversuch zeigt insgesamt, dass die Wahl der Vorverarbeitungs- und Bildverbesserungsschritte einen entscheidenden Einfluss auf die Genauigkeit des Klassifikators hat, aber auch, dass die Wahl einer gewissen Sorgfalt bedarf und Expertenwissen bzw. zeitaufwändige Tests voraussetzt. Auch zeigt der Versuch das mögliche Potential von Vorverarbeitung

insbesondere, wenn Abstufungen der Stärke oder Kombinationen von einzelnen Methoden möglich sind.

Analyse der gewonnenen Bilder

Das aSV-Verfahren wird jeweils dreimal auf jedem Datensatz wiederholt. Die gefundene Vorverarbeitung ist für die Datensätze NEU, CIFAR-10 und SVHN reproduzierbar. Daher wird für diese Datensätze der Mittelwert der Verfahren angegeben. Für RSDDs ergeben sich zwei verschiedenartige Lösungen. Die quantitativen Ergebnisse sind in Tab. 4.2 dargestellt. Beispiele für die erzeugten Bilder finden sich im Anhang (s. Abb.A.3).

Tabelle 4.2 Die Tabelle zeigt die sich ergebenden Werte für $100 \cdot \theta_{\mathrm{V},i}$ und $100 \cdot \gamma_{\mathrm{V},i}$ des aSV-Verfahrens. Eine Zahl in Klammern indiziert, dass aufgrund eines Mischfaktors $\gamma_{\mathrm{V},i} = 0$ das Originalbild verwendet wird. Zweistellige Anteile in Prozent sind fett gedruckt.

	CIFAR 10		SVHN		NEU		RSDDs (a)		RSDDs (b)	
	$\theta_{\mathrm{V},i}$	$\gamma_{\mathrm{V},i}$	$\theta_{\mathrm{V},i}$	$\gamma_{\mathrm{V},i}$	$\theta_{\mathrm{V},i}$	$\gamma_{\mathrm{V},i}$	$\theta_{\mathrm{V},i}$	$\gamma_{\mathrm{V},i}$	$\theta_{\mathrm{V},i}$	$\gamma_{\mathrm{V},i}$
-	0	-	1	-	9	-	1	-	**10**	-
HE	0	100	0	100	2	20	**15**	53	0	0
HS	0	64	0	17	(1)	0	0	33	0	99
CL	0	0	1	100	8	49	(1)	0	0	0
HF	0	100	0	0	(1)	0	0	28	0	84
BS	**93**	100	0	0	**51**	83	(1)	0	0	0
H1	7	100	**47**	100	**20**	57	**16**	82	**41**	100
H2	0	100	**33**	100	(1)	0	1	16	0	0
RF	0	0	0	0	(1)	0	**14**	94	0	100
MF	0	0	0	0	(1)	0	**33**	94	0	99
GF	0	0	0	0	(1)	0	3	75	**16**	100
MH	0	0	**18**	100	(2)	0	**10**	85	0	100
BF	0	100	0	0	(2)	0	5	79	**33**	100

Beim CIFAR-10-Datensatz hat das aSV-Verfahren eine Kombination aus ca. 93% Bildverschärfung und 7% Homogenisierung ersten Grades ermittelt. Einerseits ist dieses Ergebnis aufgrund des verschlechternden Einflusses dieser beiden Methoden (s. Tab. 4.1) unerwartet. Andererseits weisen die verarbeiteten Bilder (s. Abb. A.3 im Anhang) einen deutlich höheren optischen Schärfeeindruck auf. So sind Kanten wie bspw. der blaue

Streifen des Polizeiautos, Hintergrundgrundstrukturen und Objektgrenzen besser erkennbar. Aufgrund der Homogenisierung ist die Sättigung geringer. Die Bilder des approximierten aSV-Verfahrens unterscheiden sich nicht von denen des aSV-Verfahrens, da hier keine Parameter approximiert werden.

Der Einfluss der Vorverarbeitung bei SVHN ist deutlicher. Die drei dominierenden Bildverbesserungsmethoden sind Homogenisierung erster (47%) und zweiter (33%) Ordnung sowie der Marr-Hildreth-Operator (18%). Diese Verfahren sorgen dafür, dass die Bilder einen ähnlichen Kontrast und eine ähnliche Helligkeit haben und die Kanten hervorgehoben werden. Die Bilder verlieren ferner stark an Sättigung bzw. an Farbinformationen, was aber nicht den Informationsgehalt der Bilder reduziert, da die relevante Information in den Konturen steckt. Die Reduktion der Farbinformation sowie die Kontrast- und Helligkeitsanpassung sind hier gemäß [137] als normalisierender Weg der Konstruktion invarianter Merkmale interpretierbar. Auch hier ist kein nennenswerter Unterschied zwischen approximierten und nicht approximierten Bildern feststellbar.

Das aSV-Verfahren findet für den NEU-Datensatz ähnlich wie bei CIFAR-10 Bildverschärfung und Homogenisierung als dominante Vorverarbeitungsschritte. Ihre Anteile mit 51% (BS) und 20% (H1) unterscheiden sich aber deutlich. Darüber hinaus existiert ein nicht irrelevanter Anteil am Originalbild (9%) und an CLAHE (8%). Diese Verfahren reduzieren die in diesem Datensatz vorliegenden Beleuchtungsschwankungen mitunter erheblich und erhöhen den Kontrast bzw. verstärken die Kanten im Bild. Verglichen mit dem Originalbild sind die Texturen der Defekte der vorverarbeiteten Bilder in Abb. A.3 wesentlich deutlicher zu erkennen. Die approximierten Bilder sind mit den nicht approximierten quasi identisch, da mit BS, H1 und CL alle relevanten Operationen verwendet werden (das Originalbild zählt nicht als eine eigene Operation bei der Approximierung).

Beim RSDDs-Datensatz werden zwei verschiedene Kombinationen gefunden, die im Folgenden getrennt betrachtet werden. In beiden Fällen versucht das aSV-Verfahren, die Bilder zu homogenisieren (H1), zu glätten und gleichzeitig die Kanten zu erhalten. Im Fall (a), welcher nicht als Bild in Abb. A.3 dargestellt ist, wird eine Glättung durch das Rechteckfilter (14%), das Medianfilter (33%), das Gaußfilter (3%) und das

Bilateralfilter (5%) erreicht. Die Kanten werden durch den Marr-Hildreth-Operator (10%) und das Bilateralfilter (5%) verstärkt. Die Approximation ähnelt dem Originalbild stärker, da ein Anteil von 35% der Operationen an das Original fällt. Im Fall (b) dagegen überwiegen die Verfahren Homogenisierung ersten Grades (41%), Gaußfilterung (16%) und Bilateralfilter (33%). Ein Problem des RSDDs-Datensatzes ist, dass kleine, meist punktförmige Defekte nicht als Fehler annotiert worden sind. Die gefundene Vorverarbeitung ist darauf ausgerichtet, kleine Fehler durch einen Tiefpass zu verschleiern und gleichzeitig die Kanten zu erhalten, damit größere Defekte erkannt werden können, bei gleichzeitiger Erhöhung des Kontrastes und gleichmäßigerer Beleuchtung.

Im Folgenden werden die Ergebnisse des aSVST-Verfahrens beschrieben. Die Zusammensetzungen sind in Tab. 4.3 aufgeführt. Beispiele der gefundenen Bilder sind in Abb. A.3 im Anhang illustriert.

Tabelle 4.3 Die Tabelle zeigt die sich ergebenden Werte für $100 \cdot \theta_{\mathrm{V},i}$ und $a_{\mathrm{V},i}$ des aSVST-Verfahrens. Zweistellige Anteile in Prozent sind fett gedruckt. Irrelevante Werte für $a_{\mathrm{V},i}$ werden in der Tabelle weggelassen.

	CIFAR 10		SVHN		NEU		RSDDs	
	$\theta_{\mathrm{V},i}$	$a_{\mathrm{V},i}$	$\theta_{\mathrm{V},i}$	$a_{\mathrm{V},i}$	$\theta_{\mathrm{V},i}$	$a_{\mathrm{V},i}$	$\theta_{\mathrm{V},i}$	$a_{\mathrm{V},i}$
-	0	-	0	-	0	-	0	-
HE	0	-	4	-	0	-	5	-
HS	0	-	0	-	0	-	**34**	-
CL	0	-	0	-	**63**	1,62	0	-
HF	0	-	0	-	0	-	3	1,84
BS	**91**	6,00	0	-	**31**	3,15	0	-
H1	9	3,71	**57**	6,92	6	1,82	**14**	4,07
H2	0	-	**11**	2,26	0	-	1	1,46
RF	0	-	0	-	0	-	0	-
MF	0	-	0	-	0	-	**25**	-
GF	0	-	0	-	0	-	0	-
MH	0	-	**28**	-	0	-	**16**	-
BF	0	-	0	-	0	-	2	3,58

Insgesamt zeigen die Ergebnisse der drei Wiederholungen pro Datensatz jeweils eine hohe Reproduzierbarkeit. Lediglich die Stärkepa-

rameter $a_{\mathrm{V},i}$ schwanken etwas. Für CIFAR-10 ergibt sich eine ähnliche Zusammensetzung wie beim aSV-Verfahren, bei der Bildverschärfung das dominante Verfahren ist und ein kleiner Anteil von Homogenisierung ersten Grades vorhanden ist. Infolgedessen unterscheiden sich die produzierten Bilder auch für die approximative Variante kaum von den Bildern des aSV-Verfahrens.

Beim SVHN-Datensatz sind aSV und aSVST ebenfalls sehr ähnlich. In beiden Varianten sind Homogenisierung ersten und zweiten Grades sowie der Marr-Hildreth-Operator die dominanten Verfahren. Lediglich die Verteilung der Anteile unterscheidet sich, was zu etwas höherem Bildrauschen und einer geringen Farbsättigung führt. Durch die Approximation ändern sich die verarbeiteten Bilder kaum.

Das aSVST-Verfahren verhält sich beim NEU-Datensatz ähnlich wie beim SVHN-Datensatz. Die relevanten Verfahren sind CLAHE, Bildverschärfung und Homogenisierung ersten Grades. Hier unterscheiden sich jedoch die Anteile noch deutlicher, was sich vor allem in einem Anteil von CLAHE von ca. 63% (aSV: 8%) äußert. Hier zeigt sich der Vorteil des *Straight-Through*-Schätzers, der eine Kachelgittergröße von 3 findet, die sich deutlich von der Kachelgittergröße 8 des aSV-Verfahrens unterscheidet. Dies führt dazu, dass CLAHE vom aSVST-Verfahren als deutlich geeigneter für die Vorverarbeitung bewertet wird. Im Vergleich zu aSV haben die erzeugten Bilder eine gleichmäßigere Beleuchtung und die Texturen sind ausgeprägter. Jedoch wirken diese auch verrauschter. Durch die Approximation ändern sich die verarbeiteten Bilder ebenfalls kaum.

Die gefundene Vorverarbeitungsstrategie für RSDDs von aSVST ähnelt der Strategie (a) des RSDDs-Verfahrens. Die erzeugten Bilder beider Verfahren sind nahezu identisch. Die Zusammensetzung unterscheidet sich darin, dass statt einer Histogrammegalisierung eine Histogrammspreizung und keine Rechteckfilterung verwendet wird. Es handelt sich im Wesentlichen um eine Substitution ähnlicher Verfahren. Die approximierten Bilder sind hier unschärfer.

Insgesamt führt die Verwendung eines *Straight-Through*-Schätzers dazu, dass durch die Gradientenbildung die Parameter besser an den Datensatz angepasst werden. Dies führt dazu, dass sich einzelne Verfahren besser für einen Datensatz eignen als beim aSV-Verfahren und dass sich

die Anteile innerhalb der Vorverarbeitungsstrategien verschieben oder ähnliche Verfahren miteinander substituiert werden.

Quantitative Analyse

Im Folgenden werden die Auswirkungen der Vorverarbeitung auf die erzielten Testgenauigkeiten nach dem finalen Training des CNN beschrieben. Die Ergebnisse sind in Tab. 4.4 gegeben.

Tabelle 4.4 Erzielte Genauigkeit der Originaldatensätze und der verschiedenen vorverarbeiteten Datensätze.

	Original	aSV	aSV (appr.)	aSVST	aSVST (appr.)
CIFAR-10	93,90	94,13	94,15	94,35	**94,38**
SVHN	97,23	97,21	97,20	**97,26**	97,25
NEU	98,93	**99,87**	99,82	99,76	99,81
RSDDs	98,88	**99,07**	99,00	98,98	99,01

Für alle Datensätze kann eine Verbesserung gegenüber dem Originaldatensatz festgestellt werden. Die erzielte Genauigkeit der approximierten Variante unterscheidet sich von der nicht approximierten Variante nur unwesentlich. Bei den weniger komplexen Datensätzen (NEU, RSDDs) ist das aSV leicht besser, während bei den komplexeren Datensätzen das aSVST-Verfahren besser geeignet ist.

Die Verbesserung für den NEU-Datensatz sticht besonders hervor, da der Testfehler von $1{,}07\%$ auf $0{,}26$ bis $0{,}13\%$ gesenkt wird. Rein optisch verbessert sich der RSDDs-Datensatz durch die Vorverarbeitung zwar deutlich, aber aufgrund der teilweise inkonsistenten Annotierung (s. Abschnitt 4.1) kann das Verfahren keine derart große Verbesserung der Genauigkeit erreichen.

Beim SVHN-Datensatz dagegen liegen keine deutlichen Verbesserungen vor. Die erzielte Genauigkeit liegt unter der der Vorversuche. Eine Rolle hierbei spielt einerseits, dass durch den Unterschied zwischen Lern- und Testdaten (s. Abschnitt 4.1) keine großen Verbesserungen zu erwarten sind. Andererseits wurde das VGG-16-Netz, das in der ersten Trainingsphase verwendet wird, auf *ImageNet* [32] vortrainiert. Dieser Datensatz ist am ehesten mit CIFAR-10 zu vergleichen. Aufgrund Orthogonalität zu derartigen Datensätzen (vgl. [30]) und der Komplexität

bzgl. der Größe und den Variationen von SVHN ist die Wahl der *ImageNet*-Gewichte nicht optimal. Infolgedessen sucht das aSV-Verfahren nach einer möglichst einfachen Repräsentation der Bilddaten, um mithilfe der *ImageNet*-Merkmale des CNN den SVHN-Datensatz bestmöglich zu klassifizieren. Die suboptimale Wahl der vortrainierten Gewichte haben bei NEU und RSDDs einen geringeren Einfluss, da die beiden Datensätze eine deutlich geringere Komplexität aufweisen, weswegen diese beiden Datensätze mit allgemeineren CNN-Merkmalen klassifizierbar sind.

Fazit

Das vorgestellte Verfahren sucht mithilfe eines vortrainierten CNN nach einer gut geeigneten Vorverarbeitungsstrategie, um die Genauigkeit eines CNN-Klassifikators in einer zweiten Trainingsphase zu verbessern. Das Verfahren benötigt kein Expertenwissen und ist verglichen mit einer *brute-force*-Suche deutlich schneller. Das Verfahren ist in der Lage, die Klassifizierbarkeit des Datensatzes zu erhöhen – sogar dann, wenn das Verfahren approximiert wird. Interessanterweise sind die gefundenen Bildverbesserungsverfahren mitunter andere als jene, welche einzeln in den Vorversuchen die besten Ergebnisse liefern.

Während das aSV-Verfahren für die weniger komplexen industriellen Datensätze ausreichend ist, kann durch die Erweiterung mittels eines *Straight-Through*-Schätzers die Genauigkeit bei komplexeren Datensätzen stärker verbessert werden. Dieser Schätzer führt dazu, dass sich die Anteile der Bildverbesserungsverfahren in der Vorverarbeitungsstrategie ändern bzw. die Verfahren mit ähnlichen anderen Verfahren substituiert werden. Außerdem ist für aSVST keine Hyperparametersuche nach $\mathbf{a}_{\mathrm{V},i}^{\max}$ erforderlich. Ferner sind die Ergebnisse mit *Straight-Through*-Schätzer konsistenter. Daher wird im Folgenden das aSVST-Verfahren verwendet.

4.3.2 Augmentierung

Im Folgenden wird das FAA-Verfahren und seine Erweiterungen FAAR und FAARO aus Abschnitt 3.3 sowohl für die Originaldatensätze als auch für die vorverarbeiteten Datensätze (vgl. Abschnitt 4.3.1) evaluiert. Hierfür wird eine frei zugängliche `pytorch`-Implementierung für `python`, die entsprechend erweitert wurde, mit den Standardparametern verwendet. Für jeden Datensatz wird die GAN-Struktur des FAA-Verfahrens

mit 20 Durchläufen trainiert. Jedes Experiment wird dreimal wiederholt. Anschließend wird für die quantitative Analyse mit der gefundenen Augmentierungsstrategie ein VGG-16-Netz für 100 Durchläufe trainiert. Hierfür wird wie in den Experimenten zuvor wieder eine `tensorflow`-Umgebung verwendet. Eine Augmentierungsstrategie $\mathcal{T}_{\text{aug}}$ besteht in dieser Arbeit aus zehn Unterstrategien mit jeweils zwei Operationen.

Quantitative Analyse

Die erzielten Genauigkeiten für die Augmentierungsverfahren bei den Originaldatensätzen und den verschiedenen vorverarbeiteten Datensätzen sind in Tab.4.5 aufgeführt.

Tabelle 4.5 Erzielte Testgenauigkeit der Augmentierungsverfahren (Augm.) bei den Originaldatensätzen und den vorverarbeiteten Datensätzen (aSVST). Die Basis-Augmentierung enthält Spiegelung und *Cutout*.

Augm.	VV.	CIFAR-10	SVHN	NEU	RSDDs
Basis	Original	93,90	97,29	98,93	98,88
FAA		93,50	97,31	98,72	98,59
FAAR		94,60	97,34	98,80	98,94
FAARO		95,08	**97,44**	99,08	99,16
Basis	aSVST	94,31	97,26	**99,74**	98,93
FAA		94,70	97,28	98,33	99,11
FAAR		94,73	97,37	99,26	**99,32**
FAARO		**95,23**	97,42	99,02	99,24

Beim CIFAR-10-, SVHN- und RSDDs-Datensatz tendieren CNNs, deren Training mit stärker modifizierten FAA-Verfahren oder Vorverarbeitung durchgeführt wurde, zu einer besseren Klassifikationsleistung. Dies entspricht einerseits den Ergebnissen aus Abschnitt 4.3.1 und andererseits den Erwartungen an die Modifikationen des FAA-Verfahrens (s. Abschnitt 3.3.2). Somit führt eine gut geeignete Vorverarbeitung und Augmentierung bei CIFAR-10 gleichermaßen zu einer Verbesserung der Genauigkeit.

Beim SVHN-Datensatz ist die Auswirkung der Vorverarbeitung auch bei unterschiedlich modifizierter Augmentierung gering und spielt eine untergeordnete Rolle. Durch Augmentierung kann eine geringfügige Verbesserung der Genauigkeit von 0,15 Prozentpunkten erreicht werden.

Der RSDDs-Datensatz verhält sich ähnlich wie der CIFAR-10-Datensatz. Jedoch erreicht das FAAR-Verfahren bei Vorverarbeitung leicht bessere Ergebnisse als das FAARO-Verfahren.

Beim NEU-Datensatz verschlechtert sich die Genauigkeit bei Augmentierung. Lediglich der Originaldatensatz mit FAARO führt zu einer Verbesserung. Gleichzeitig können Fehlklassifikationen durch die aSVST-Vorverar-beitung um rund 76% reduziert werden. Hier zeigen sich die Schwächen des FAA-Verfahrens.

Die Augmentierung während des FAA-Verfahrens unterscheidet sich von der gefundenen Augmentierungsstrategie. Dies liegt daran, dass während des Trainings eine gewichtete Summe aus allen zur Verfügung stehenden Operationen $O_{i,\mathrm{AA}}$ gebildet wird und die Unterstrategien nach dem Training aus einer Stichprobe entnommen werden, wobei die Wahrscheinlichkeiten den Gewichten der jeweiligen Operation entsprechen. Dies kann zu suboptimalen Strategien führen. Ein weiterer Grund ist speziell beim NEU-Datensatz die hohe Ähnlichkeit einiger Klassen, während innerhalb der Klassen eine mitunter große Varianz vorherrscht. Bei RSDDs liegt ein ähnliches Problem vor, da teilweise Defekte nicht als solche annotiert sind. Beim Augmentieren können durch bestimmte Operationen klassenspezifische Charakteristika nivelliert werden. Ferner tritt bei allen Varianten des NEU-Datensatzes Überanpassung auf, weshalb sich durch die Augmentierung die Generalisierungslücke Γ vergrößert.

Analyse der gefundenen Strategien

Da die Unterstrategien einer Stichprobe aus einer verallgemeinerten Ber-noulli-Verteilung entstammen, unterscheiden sich die gefundenen Augmentierungsstrategien bei jedem Durchlauf geringfügig. Um die gefundenen Augmentierungsstrategien besser unterscheiden zu können, werden die Operationen in fünf verschiedenen Kategorien klassifiziert, welche grob der Einteilung aus Abschnitt 3.3 entsprechen. Diese sind Koordinatentransformationen (Scherung, Translation, Rotation, Spiegelung, zufälliger Zuschnitt), Farbmanipulation (Invertierung, Solarisation, Kontrast-, Helligkeits-, Sättigungsmanipulation, Posterisation, Farbverschiebung), normalisierende Verfahren (Autokontrast, Histogrammegalisierung), Schärfe (Bildverschärfung, Gaußfilterung) und das Hinzufügen von Rauschen (Impulsrauschen, normalverteiltes Rauschen, gleichver-

teiltes Rauschen). Insgesamt werden in den drei Wiederholungen pro Datensatz 60 Operationen ermittelt. Untersucht wird nur das FAARO-Verfahren. Die Ergebnisse sind in Tab. 4.6 zusammengefasst.

Tabelle 4.6 Anzahl der gefundenen Operationen unterteilt in Kategorien für das FAARO-Verfahren bei den Originaldatensätzen und den vorverarbeiteten Datensätzen (aSVST). Die Ergebnisse in Klammern beim NEU-Datensatz beziehen sich auf ein Experiment ohne Operationen der Normalisierung und ohne Invertierung. Die Kategorien sind Koordinatentransformationen (affine Transformation, Spiegelung etc.), Farbmanipulationen (Helligkeit, Sättigung, Kontrast etc.), normalisierende parameterfreie Verfahren wie Histogrammegalisierung, das Hinzufügen von Rauschen und Schärfeoperationen (Gaußfilterung, Bildverschärfung).

Kategorie	VV.	CIFAR-10	SVHN	NEU	RSDDs
Koordinaten		18	20	33	23
Farbe		16	11	16	16
Normalisierung	Original	8	26	2	15
Rauschen		2	0	3	3
Schärfe		16	3	6	3
Koordinaten		18	39	20 (26)	21
Farbe		13	13	13 (23)	19
Normalisierung	aSVST	5	5	20 (-)	0
Rauschen		14	2	3 (8)	7
Schärfe		10	1	4 (3)	13

Interessant ist hierbei der Vergleich zwischen dem Originaldatensatz und den mit aSVST vorverarbeiteten Daten. Beim CIFAR-10-Datensatz sinken die Anteile von Farbmanipulation, Normalisierung und Schärfe zu Gunsten von Rauschen durch die aSVST-Vorverarbeitung. Diese Veränderung ist auf die hohen Anteile von Bildverschärfung und Homogenisierung zurückzuführen, die die Bilder hinsichtlich visuellem Schärfeeindruck, Farbe, Kontrast und Helligkeit teilweise normalisieren. Durch die Bildverschärfung wird Rauschen verstärkt, daher versucht das FAARO-Verfahren, dagegen Invarianzen aufzubauen. Dies führt dazu, dass normal- und gleichverteiltes Rauschen sowie die Gaußfilterung, die Rauschen abmildert, einen hohen Anteil an der Augmentierungsstrategie haben, was insgesamt zu der höchsten erreichten Genauigkeit für CIFAR-10 führt (vgl. Tab. 4.5).

Der Einfluss auf die Genauigkeit bei SVHN durch aSVST ist gering. Dennoch ändert sich bei der Verwendung von aSVST $\mathcal{T}_{\text{aug}}$ erheblich. Am auffälligsten ist die Verschiebung von normalisierenden Operationen hin zu Koordinatentransformationen aufgrund der starken Homogenisierung bei aSVST. Dies weist darauf hin, dass nach der Verarbeitung mit aSVST eine deutlich geringe Notwendigkeit für normalisierende Operationen besteht und Invarianten bzgl. affinen Transformationen in den Fokus rücken.

Beim RSDDs-Datensatz ist die deutliche Verschiebung von normalisierenden Operationen zu Schärfeoperationen und Rauschen erkennbar. Da bei aSVST die Kontrastnormalisierung bereits durch Histogrammspreizung und Homogenisierung abgebildet wird, versucht das FAARO-Verfahren, Invarianzen gegenüber Bildschärfe und Rauschen aufzubauen. Hierbei wird versucht, kleine, nicht annotierte Fehler nachzubilden.

Die verschiedenen FAA-Verfahren führen beim originalen NEU-Datensatz zwar zu einer Verbesserung, verglichen mit dem vorverarbeiteten Datensatz ist aber eine deutliche Verschlechterung festzustellen. Dies liegt insbesondere daran, dass das FAARO-Verfahren beim vorverarbeiteten Datensatz einen hohen Anteil an kontrastnormalisierenden Verfahren auswählt, welche die klassenspezifischen Charakteristika im Bild zerstören (s. Abb. 4.6). Dadurch sinkt die Fähigkeit des CNN zu generalisieren. Stattdessen werden bestimmte Bildeigenschaften auswendig gelernt, wodurch die Generalisierungslücke Γ steigt. Dieser destruktive Einfluss ist in anderen Datensätzen nicht feststellbar, da die Histogrammegalisierung primär die Textur verändert, die in dieser Stärke nur beim NEU-Datensatz maßgeblich für die Klassenzugehörigkeit ist.

In einem weiteren Versuch wird das FAARO-Verfahren ohne die normalisierenden Operationen (Histogrammegalisierung, Auto-Kontrast) sowie ohne Invertierung durchgeführt. Damit werden keine parameterfreien Methoden verwendet, die die Klassenzugehörigkeitsinformationen schädigen können oder zu unerwarteten Effekten bei der Anwendung führen. Die Ergebnisse hierzu sind in Tab. 4.6 in der Spalte des NEU-Datensatzes in Klammern aufgeführt. Die damit erzielte Genauigkeit liegt bei über 99,99% und ist damit die höchste erzielte Genauigkeit dieses Datensatzes (vgl. Tab. 4.5). Im Folgenden wird das FAARO-Verfahren für NEU immer ohne normalisierende Verfahren verwendet.

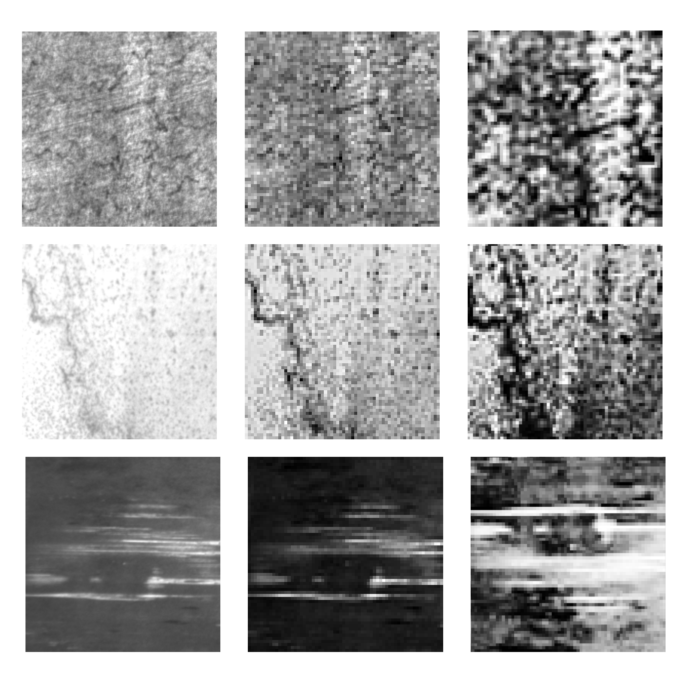


Abbildung 4.6 Drei Beispiele für die augmentierten und vorverarbeiteten Bilder von unterschiedlichen Klassen des NEU-Datensatzes. Die Unterstrategie besteht jeweils aus Histogrammegalisierung und zufälligem Ausschneiden. Die linke Spalte zeigt jeweils das Originalbild, die mittlere das mit aSVST verarbeitete Bild und die rechte das augmentierte Bild. Während die Klasse der Bilder in ersten beiden Spalten gut unterscheidbar ist, fällt die Unterscheidung bei der dritten Spalte zu den oberen beiden Beispielen schwer. Durch die Histogrammegalisierung werden hier die spezifischen Charakteristika der Klassen verringert.

Bewertung des Ressourcenbedarfs

Im Folgenden wird der Einfluss von Vorverarbeitung und Augmentierung auf den Ressourcenbedarf des CNN untersucht. Der Ressourcenbedarf wird hierzu implizit mithilfe des *Channel-Pruning*-Verfahrens [64] (vgl. Abschnitt 3.4.3) bestimmt. Je redundanter die Strukturen nach dem Training sind, desto weniger Gewichte werden effektiv für die Klassifikation benötigt und desto mehr Gewichte werden entfernt. Dies führt zu einem geringeren Ressourcenbedarf in der Inferenz. Das verwendete VGG-16-Netz hat insgesamt 4.224 Filterkerne $\mathbf{k}_l$ und $|\mathcal{H}| = 14.736.714$

Gewichte (für NEU und RSDDs geringfügig weniger aufgrund der geringeren Klassenanzahl).

Tabelle 4.7 Anzahl der Filterkerne $\mathbf{k}_l$, die durch das *Channel-Pruning*-Verfahren entfernt werden.

Augm.	VV.	CIFAR-10	SVHN	NEU	RSDDs
FAARO	Original	2.631	2.919	2.944	3.350
Basis	aSVST	2.715	3.045	2.638	3.494
FAARO	aSVST	2.705	2.928	2.857	3.394

Tabelle 4.8 Anzahl der Gewichte $|\mathcal{H}|$ in Tausend des CNN nach dem *Channel-Pruning*-Verfahren.

Augm.	VV.	CIFAR-10	SVHN	NEU	RSDDs
FAARO	Original	1.303	621	1.116	13.660
Basis	aSVST	1.132	447	2.572	127
FAARO	aSVST	1.162	594	1.407	52

In Tab. 4.7 sind die durch das *Channel-Pruning* entfernten Filterkerne und in Tab. 4.8 ist die Anzahl der verbleibenden Gewichte aufgeführt. Für CIFAR-10, SVHN und RSDDs führt die Vorverarbeitung mit Basisaugmentierung verglichen mit dem FAARO-Verfahren ohne Vorverarbeitung zu einer höheren Redundanz innerhalb des jeweiligen CNN. In der Folge werden mehr Filterkerne entfernt, wodurch weniger Gewichte in der Inferenz benötigt werden. Werden die vorverarbeiteten Bilder auch mit FAARO augmentiert, steigt die Anzahl der Gewichte leicht bzw. reduziert sich weiter für RSDDs. Da beide augmentierenden Verfahren eine ähnliche Genauigkeit aufweisen (s. Tab. 4.5), zeigt dies, dass die Vorverarbeitung mittels aSVST-Verfahren eine ressourcensparende Alternative zu starker Augmentierung sein kann.

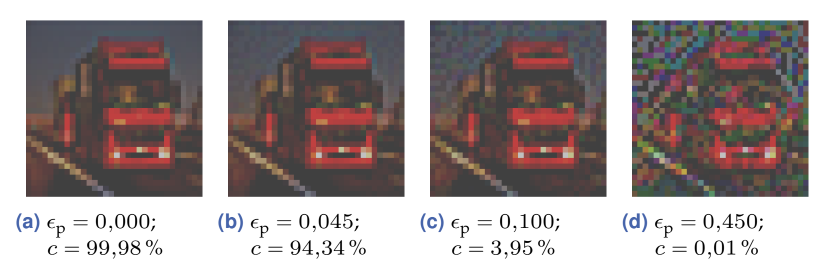
Beim NEU-Verfahren führt die Vorverarbeitung zu einer überdurchschnittlichen Steigerung der Testgenauigkeit. Dies rechtfertigt eine geringere Redundanz im CNN und daher ist auch kein fairer Vergleich mit FAARO hinsichtlich der Parameter möglich. Durch die Augmentierung des vorverarbeiteten NEU-Datensatzes sinkt die Anzahl der Parameter

weiter, während die Testgenauigkeit sich dem Wert 1 annähert. Dieses Verhalten ist auch bei RSDDs feststellbar.

Insgesamt sind durch das aSVST-Verfahren somit zwei Verhaltensweisen feststellbar. Für relativ große Datensätze (CIFAR-10, SVHN) führt die Vorverarbeitung zu einem geringeren Ressourcenanspruch an das CNN, während die Klassifikationsgenauigkeit geringfügig steigt. Bei kleinen Datensätzen, die teilweise unter schlechten Aufnahmebedingungen erstellt wurden, führt die Vorverarbeitung zu deutlichen Verbesserungen bzgl. der Genauigkeit. Dies kann aber auch mit einer Steigerung der benötigten Ressourcen einhergehen.

Robustheit gegenüber *Adversarial Attacks*

Als *Adversarial Attacks*, zu Deutsch feindliche Angriffe, werden Angriffe auf einen CNN-Klassifikator bezeichnet, die mit scheinbar minimaler Änderung des Eingabebildes eine Fehlklassifikation des CNN provozieren. Im Folgenden wird die Robustheit von CNNs, die mit vorverarbeiteten bzw. augmentierten Daten trainiert wurden, gegenüber zwei konträren Angriffe untersucht, die eine Störung abhängig vom jeweiligen CNN und seiner Gewichte $\mathcal{H}$ sowie des Eingabebildes $\mathbf{X}$ berechnen.



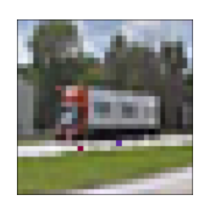
(a) $\epsilon_\text{p} = 0{,}000$; $c = 99{,}98\,\%$ (b) $\epsilon_\text{p} = 0{,}045$; $c = 94{,}34\,\%$ (c) $\epsilon_\text{p} = 0{,}100$; $c = 3{,}95\,\%$ (d) $\epsilon_\text{p} = 0{,}450$; $c = 0{,}01\,\%$

Abbildung 4.7 Beispiel aus dem CIFAR-10-Datensatz [91] für einen Angriff mit linearer Perturbation. Die Abbildung zeigt ein Bild der Klasse LKW bei unterschiedlich starken Perturbationen und die erzielte Konfidenz c des Klassifikators für diese Klasse. Die Konfidenz sinkt für stärkere Perturbationen stark, was final zur Fehlklassifikation führt, obwohl der LKW noch für $\epsilon_\text{p} = 0{,}450$ erkennbar ist.

Beim Angriff mit linearer Perturbation nach [52] wird die Schwäche von CNNs ausgenutzt, dass diese u. a. aufgrund ihrer ReLU-Aktivierungen *zu linear* sind, was einerseits das Training durch die bekannten Gradientenabstiegsverfahren beschleunigt, aber es andererseits einfa-

chen linearen Störungen des Eingabebildes ermöglicht, die Klassifikation stark negativ zu beeinflussen. Diese Annahme fußt auf der Beobachtung, dass die Zielfunktion eines CNN über Abwandlungen von $\nabla_{\mathcal{H}} f(\mathcal{X}_L, \mathcal{H})$ minimiert wird. GOODFELLOW et al. schlagen in [52] vor, mithilfe des Gradienten bzgl. eines Eingabebildes $\mathbf{X} \in \mathcal{X}_L \vee \mathcal{X}_T$ eine perturbative Störung $\Xi = \epsilon_p \text{sign}(\nabla_{\mathbf{X}} f(\mathbf{X}, \mathcal{H}))$ zu bestimmen. Die Stärke der Störung ist über ϵ_p anpassbar. Für die Angriffe wird die Störung Ξ auf das Eingabebild addiert und in das CNN eingegeben. Ein Beispiel für einen solchen Angriff ist in Abb. 4.7 dargestellt.

Statt eines Rauschmusters, das die Pixelwerte des gesamten Bildes betrifft, werden beim zweiten Angriff nur einzelne Pixelwerte verändert. Das in [152] vorgestellte Verfahren sucht mittels differentieller Evolution nach der Position und dem Farbwert, bei dem die Konfidenz des CNN für die wahre Klasse des Bildes minimal wird. Im Erfolgsfall führt dies zur Fehlklassifikation des jeweiligen Bildes. Hierbei wird angenommen, dass durch diese Angriffe das gestörte Bild außerhalb der intrinsischen Verteilung des Trainingsdatensatzes liegt. Beispiele aus dem CIFAR-10 Datensatz, die einen erfolgreichen Angriff zeigen, sind in Abb. 4.8 illustriert.


(a) Flugzeug (LKW)


(b) Flugzeug (Katze)

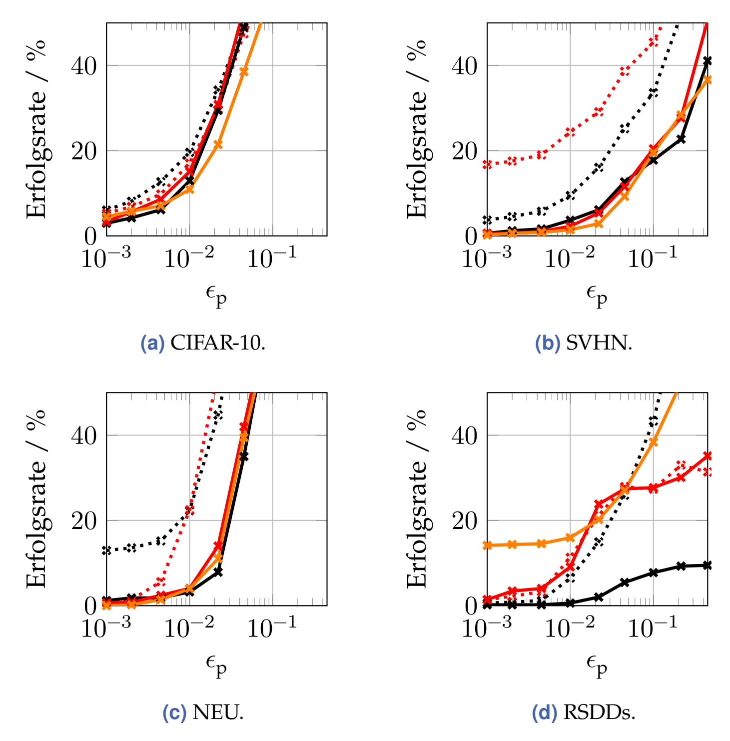

(c) Flugzeug (Vogel)


(d) LKW (Schiff)

Abbildung 4.8 Beispiele aus dem CIFAR-10-Datensatz [91] für einen Pixelangriff. Die Bildunterschriften geben die geschätzte Klasse an, während in Klammern die wahre Klasse angegeben wird.

Beide Angriffe werden auf jeweils 500 Testbildern der jeweiligen Datensätze ausgeführt. Aufgrund numerischer Schwierigkeiten bei einer Überkonfidenz [142] werden die Gradienten anhand der Ausgabe vor der letzten Aktivierung berechnet. Darüber hinaus werden nur Bilder betrachtet, die korrekt klassifiziert werden.

Die Erfolgsraten für den Angriff mit Rauschmuster sind Abb. 4.9 dargestellt. Die Abbildungen deuten darauf hin, dass durch eine geeignete



(a) CIFAR-10. **(b)** SVHN. **(c)** NEU. **(d)** RSDDs.

Abbildung 4.9 Erfolgsraten der Rauschmusterangriffe für die vier Datensätze. Durchgezogene Linien weisen auf die Verwendung des aSVST-Verfahrens hin, während bei gepunkteten Linien keine Vorverarbeitung angewandt wurde. Bei schwarzen Linien wird die Basisaugmentierung und bei roten FAARO-Augmentierung verwendet. Bei der orangen Linie wird eine spezielle Rauschaugmentierung verwendet.

Vorverarbeitung die Erfolgsrate des Angriffs deutlich reduziert werden kann. Der Einfluss von Augmentierung ist hier deutlich geringer, führt aber auch zu einer leicht steigenden Resilienz gegenüber diesem Angriff. Beim NEU-Datensatz führt die Verwendung von FAARO oder von aSVST für leichte Störungen fast zu einer vollständigen Robustheit. Beim SVHN-Datensatz dagegen führt FAARO allein zu einer deutlichen Verschlechterung gegenüber der Basisaugmentierung. Ursache hierfür ist der große Anteil normalisierender Operationen bei FAARO (vgl. NEU, s. Tab. 4.5). Eine Vorverarbeitung führt dazu, dass irrelevante Informatio-

nen in den Bildern reduziert werden, während relevante Informationen verstärkt werden (s. Abb. A.3). Dies führt nach dem Training zu besser diskriminierenden Merkmalen, die in der Folge robuster gegenüber gradientenbasierten Angriffen sind.

Die entsprechenden Ergebnisse für Angriffe der zweiten Art auf einen, drei und fünf Pixelwerte sind in Tab. 4.9 aufgeführt und deuten darauf hin, dass hier eine gute Augmentierung den Erfolg des Angriffs stärker reduziert als eine gute Vorverarbeitung. Auffallend ist, dass Datensätze, deren Klasseninformationen über das gesamte Bild verteilt ist, eine deutlich höhere Resilienz gegenüber dieses Angriffs haben: Während die Informationen beim NEU-Datensatz in der Textur liegen, steckt eine hohe Redundanz bzgl. der Information in den Konturen der Ziffern bei SVHN. Diese Redundanz liegt beim CIFAR-10-Datensatz nicht vor, da die Semantik im Bild stark von der Kohärenz einzelner Pixel abhängt. Durch die Veränderung weniger Pixel verändern sich die extrahierten Merkmale daher bei CIFAR-10 deutlich stärker als bei SVHN (vgl. [40]). Ebenso ist der RSDDs-Datensatz sehr anfällig, da die Information über das Vorhandensein eines Defekts einen sehr lokalen Charakter hat. Der positive Einfluss der Augmentierung rührt daher, dass der Stichprobenumfang beim Lernen größer ist, wodurch die intrinsische Statistik des Datensatzes besser durch das neuronale Netz erfasst werden kann. Je stärker die Augmentierung, desto besser ist die Erfassung möglich. Das CNN wird darüber hinaus nicht nur gezwungen, mehr zu generalisieren, sondern kann Eingabebilder, die u. U. bei schlechter Augmentierug als *nicht in der gelernten Verteilung* interpretiert werden würden, besser erkennen. Dies führt zu einer höheren Resilienz gegenüber vermeintlichen Ausreißern aus der Verteilung des Lerndatensatzes, wie sie der Pixelangriff versucht zu erzeugen.

In einem weiteren Versuch werden die vorverarbeiteten Datensätze mit Basisaugmentierung und zusätzlich vier Rauschoperationen (gleichverteiltes Rauschen, gaußverteiltes Rauschen, Impulsrauschen und Pixelrauschen entsprechend des Pixelangriffs) trainiert, um die beiden Angriffe abzuwehren. Es ist jedoch anzumerken, dass hierdurch versucht wird, zufällige Variationen zu erkennen und einzuordnen, und nicht gezielt Angriffe präventiv abzuwehren.

Tabelle 4.9 Erfolgsraten in % der Pixelangriffe für die vier Datensätze für einen, drei und fünf Pixel (px).

		-		aSVST		
	px	Basis	FAARO	Basis	FAARO	Rauschen
CIFAR	1	41,6	7,1	20,4	21,9	10,7
	3	54,2	23,0	40,1	29,5	12,9
	5	59,8	22,2	47,0	33,5	15,6
SVHN	1	16,6	6,6	11,5	6,1	2,1
	3	34,4	14,1	14,2	8,2	4,1
	5	40,2	19,4	18,3	8,3	4,7
NEU	1	1,6	<0,1	0,5	0,5	<0,1
	3	7,8	<0,1	0,5	0,5	<0,1
	5	10,6	0,5	2,5	0,5	<0,1
RSDDs	1	44,2	3,7	2,8	<0,1	<0,1
	3	71,0	9,2	5,4	<0,1	<0,1
	5	75,1	13,4	8,6	<0,1	<0,1

Während die Anfälligkeit gegenüber dem Rauschangriff sich mit Ausnahme von RSDDs, wo die Erfolgsrate stark steigt, kaum ändert (s. orange Linien in Abb. 4.9), können durch die spezielle Rauschaugmentierung Pixelangriffe abgewehrt werden. Jedoch geht dies zulasten der Genauigkeit, die verglichen mit der Basisaugmentierung (s. Tab. 4.5) bei CIFAR-10 von 94,31 % auf 92,46 %, bei SVHN von 97,26 % auf 96,75 %, bei NEU von 99,74 % auf 99,43 % und bei RSDDs von 98,93 % auf 98,71 % fällt. Dies entspricht den Beobachtungen von [82], dass eine Augmentierung gegen Angriffe zwar die Robustheit gegenüber diesen erhöht, aber nicht zum Lernen höherwertiger Merkmale beim CNN führt.

Insgesamt führt Augmentierung und Vorverarbeitung zu einer höheren Resilienz gegenüber Angriffen auf den Klassifikator. Insbesondere auf unverarbeiteten bzw. kleinen Bilddatensätzen (RSDDs und NEU) bietet aSVST und FAARO einen besonders hohen Schutz gegenüber Angriffen. Das gezielte Augmentieren mit potentiellen Angriffen verhindert nur den Erfolg von Impulsangriffen effektiv und führt zu einem Verlust der Genauigkeit nicht nur gegenüber der Augmentierungsstrategie der *AutoAugment*-Verfahren, sondern auch gegenüber der Basisaugmentie-

rung. Dieser Verlust ist ein Indiz dafür, dass das CNN mit aSVST und FAARO zu einer höheren Generalisierungsfähigkeit neigt [82].

Fazit

Dieser Abschnitt hat gezeigt, dass das FAA-Verfahren und dessen Varianten ein legitimes Verfahren für die Suche nach einer geeigneten Augmentierungsstrategie sein kann. Eine geeignete Auswahl der Operationen und die Verwendung einer stochastischen Operationsstärke hat einen positiven Einfluss auf das Verfahren: Normalisierende Operationen können aufgrund der Stichprobe, aus der am Ende die Augmentierungsstrategie entnommen wird, einen unerwarteten Einfluss auf die Texturen im Bild haben. Die Berücksichtigung von Operationen, die die Schärfe- und Rauscheigenschaften im Bild ändern, sind für manche Datensätze wie CIFAR-10 notwendig.

Augmentierung und Vorverarbeitung können beide als Verfahren zur Konstruktion invarianter Merkmale nach [137] interpretiert werden. Beide führen zu einer erhöhten Testgenauigkeit und bei einer Kombination beider Verfahren werden in dieser Arbeit die besten Genauigkeiten erzielt. Durch die Vorverarbeitung werden gewisse Invarianzen gebildet, die nicht durch die Augmentierung abgedeckt werden müssen. Daher ändert sich die gefundene Augmentierungsstrategie bei einer Kombination. Der Ressourcenbedarf, der durch das kombinierte Verfahren impliziert wird, tendiert in zwei Richtungen. Einerseits reduziert er sich für große Datensätze, da durch die normalisierende Vorverarbeitung weniger Variationen auftreten, und andererseits kann insbesondere durch die Vorverarbeitung die Genauigkeit deutlich verbessert werden, was zu einem leicht steigenden Bedarf führen kann (NEU). Ferner sind CNNs robuster gegenüber Angriffen, falls sie mit vorverarbeiteten bzw. augmentierten Bildern trainiert wurden. Jo und Bengio [82] zeigen, dass die Robustheit gegenüber Angriffen der Schlüssel für Generalisierung darstellen, da eine hohe Sensitivität dafür spricht, dass das CNN keine Sachverhalte abstrahieren kann und keine segmantischen Konzepte, sondern statistische Irregularitäten in den Datensätzen gelernt hat.

4.4 Evaluation des Quantisierungsverfahrens

In diesem Abschnitt wird das Quantisierungsverfahren aus Abschnitt 3.4 evaluiert. Hierfür wird zunächst in Abschnitt 4.4.1 eine 8-Bit-Quantisierung näher untersucht, wobei u. a. die Hyperparameter und Erweiterungen der Quantisierung analysiert werden. In Abschnitt 4.4.2 wird das Verhalten bei einer niedrigeren Bitlänge untersucht. Anschließend werden in Abschnitt 4.4.3 Stärken und Schwächen des Quantisierungsverfahrens diskutiert.

Angelehnt an die Standardparameter der Chargennormalisierung wird ein Momentum $m_{\text{q}} = 0{,}99$ gewählt. Dies ist notwendig, um in der Endphase des Trainings die Gewichte zu stabilisieren. Jedoch kann dies auch dazu führen, dass bei einer schlechten Initialisierung der Skalierungsfaktoren kein Optimum der Zielfunktion gefunden wird.

4.4.1 Ergebnisse der 8-Bit-Quantisierung

Im Folgenden wird das Quantisierungsverfahren und seine Erweiterungen mit jeweils 8 Bit untersucht. Dies ist die technisch bedeutsamste Bitlänge, da moderne Rechnersysteme ihre Speicher in Bytes organisieren. In diesem Abschnitt wird zunächst in Voruntersuchungen der Einfluss relevanter Hyperparameter analysiert. Anschließend werden die in [185] veröffentlichten Erweiterungen untersucht. Am Ende wird das vorgestellte Quantisierungsverfahren bei unterschiedlichen Netztopologien untersucht.

Voruntersuchungen

In diesem Abschnitt werden Vorversuche zur Quantisierung anhand des CIFAR-10-Datensatzes mit einem VGG-10-Netz durchgeführt. Diese umfassen die Untersuchung bzgl. der geeignetsten Verteilung der Bits zwischen Ganzzahl- und Nachkommabitlänge, dem Einfluss von Rauschen auf Aktivierungen und Gewichte und dem optimalen Puffer α_{q}. Bei jedem Experiment wird das CNN 100 Durchläufe mit dem gemäß aSVST-Verfahren verarbeiteten CIFAR-10 mit dazugehöriger FAARO-Augmentierung trainiert. Aufgrund der Vielzahl der Vorversuche wurden die Trainings nicht wiederholt. Die Ergebnisse können damit leicht je

nach Versuch variieren und dienen lediglich dazu, bestimmte Tendenzen zu ermitteln.

Tabelle 4.10 Testgenauigkeit in % für CIFAR-10 bei unterschiedlicher Verteilung der Bits mit und ohne künstlichem Quantisierungsrauschen nach der Konvertierung der Schichten und der Quantisierung der Gewichte. Testgenauigkeiten von unter 85% sind nicht aufgeführt, stattdessen wird die Ursache der schlechten Genauigkeit (Q: Scheitern durch Quantisierung der Gewichte, K: starke Konvergenzprobleme, L: leichte Konvergenzprobleme) angegeben (s. u.). Die erste Zeile gibt die Verteilung der Bits für die Gewichte als Tupel $(L_{\mathrm{q,I,W}}, L_{\mathrm{q,F,W}})$ an. Die erste Spalte gibt entsprechend die Verteilung $(L_{\mathrm{q,I,A}}, L_{\mathrm{q,F,A}})$ der Aktivierungen an.

		Verteilung der Gewichte						
		(-2,10)	(-1,9)	(0,8)	(1,7)	(2,6)	(3,5)	(4,4)
Verteilung der Aktivierungen		ohne Hinzufügen von Rauschen						
	(5,3)	91,25	93,29	(K)	(K)	(K)	(K)	(K)
	(4,4)	91,24	93,54	92,33	(K)	93,52	89,68	(K)
	(3,5)	92,36	93,68	92,91	92,62	92,17	90,52	(Q)
	(2,6)	(L)	87,13	89,57	89,62	90,57	(Q)	(Q)
	(1,7)	(L)	(L)	(L)	(Q)	(Q)	(Q)	(Q)
		mit künstlichem Quantisierungsrauschen						
	(5,3)	90,94	92,05	(K)	(K)	(K)	(K)	(K)
	(4,4)	91,98	93,21	93,57	93,88	93,11	92,12	(K)
	(3,5)	91,85	93,25	92,84	92,81	91,88	91,46	(L)
	(2,6)	(L)	87,75	87,74	90,67	90,39	(L)	(K)
	(1,7)	(L)	(L)	(L)	(K)	(K)	(K)	(K)

Die Ergebnisse der verschiedenen 8-Bit-Verteilungen mit $\alpha_{\mathrm{q}} = 1{,}1$ sind in Tab. 4.10 dargestellt. Es kann keine verlässliche Aussage darüber getroffen werden, ob das künstliche Quantisierungsrauschen zu einer Verbesserung oder Verschlechterung der Genauigkeit führt, da sich das Verfahren diesbezüglich im Einzelfall unterschiedlich verhält. Die Ursachen für das Scheitern liegen entweder im Quantisierungsschritt oder bei den schlechten Konvergenzeigenschaften während des Trainings. Während bei Ersterem ein Verlust an Genauigkeit bei der Quantisierung von betragsmäßig kleinen Gewichten entsteht, führen schlechte Konvergenzeigenschaften dazu, dass vor Konvertierung und Quantisierung (vgl. Abb. 3.5) keine Genauigkeiten von über 85 % erreicht werden. Ohne das Hinzufügen von Rauschen können die Ursachen des Scheiterns

drei verschiedenen Fällen zugeordnet werden (s. Tab. 4.10): Für grobe Aktivierungen in Kombination mit groben Gewichten treten extreme Konvergenzprobleme auf, bei denen die erzielte Genauigkeit der eines Münzwurfs entspricht (Kennzeichen (K) in Tab. 4.10). Bei einer Kombination von feineren Quantisierungsstufen der Aktivierungen mit groben Gewichten tritt ein Verlust der Genauigkeit aufgrund der Quantisierung der Gewichte auf (Q). Dies ist die Folge der näherungsweise gaußförmigen Verteilung der Gewichte um den Wert Null gepaart mit der Zuordnung aller Gewichte mit $|h_i| < \Delta_W/2$ auf den Wert Null. Je größer $L_{q,I,W}$, desto kleiner ist $L_{q,F,W}$ und desto mehr Werte werden auf null gerundet. In der dritten Region treten leichte Konvergenzprobleme auf (L), d. h. es werden Genauigkeiten zwischen $20\,\%$ und $85\,\%$ erreicht, da die Amplitude der Aktivierungen zu klein ist. Die Quantisierungsprobleme sind gering und steigen mit der Ganzzahlbitlänge der Gewichte. Der Übergang zwischen den Regionen (Q), (L) und der konvergierenden Region ist fließend. Ferner führt das künstliche Rauschen dazu, dass bei ungünstigen Bit-Kombinationen das Training seltener scheitert. Wie Tab. 4.10 zeigt, wird durch das Rauschen bei der Kombinationen $L_{q,A} = (4,4)$ und $L_{q,W} = (1,7)$ eine Testgenauigkeit von über $85\,\%$ erreicht.

Das Rauschen auf Gewichte und Aktivierungen bewirkt einerseits, dass bei jeweils groben Quantisierungen der Quantisierungseffekt berücksichtigt wird. Gleichzeitig kann es aber zu Konvergenzproblemen während des Trainings kommen, da das Rauschen einen zu starken Störeffekt auf die Schätzung des Gradienten hat. Die Region der Quantisierungsprobleme löst sich daher zu Gunsten der leichten (L) und schweren (K) Konvergenzprobleme auf.

In beiden Versuchsreihen treten jeweils die wenigsten Schwierigkeiten für $L_{q,W} = (-1,9)$ und $L_{q,A} = (3,5)$ auf. Daher wird im Folgenden die Kombination aus beidem verwendet, d. h. $L_{q,W} = (-1,9)$ und $L_{q,A} = (3,5)$. Ferner wird die Hinzugabe von künstlichem Rauschen verwendet, um Quantisierungsprobleme zu vermeiden. Eine ungünstige Wahl der Quantisierungsstufen[1] führt nicht nur zu schlechteren Testgenauigkeiten, sondern auch zu schlechten Konvergenzeigenschaften. Diese können ggf. durch Vortraining (s. u.) verbessert werden.

[1] Dieses Problem tritt auch bei tieferen CNNs und gröberer Quantisierung auf. Hinweise, woran dies liegt, werden verteilt im gesamten Abschnitt 4.4 geben.

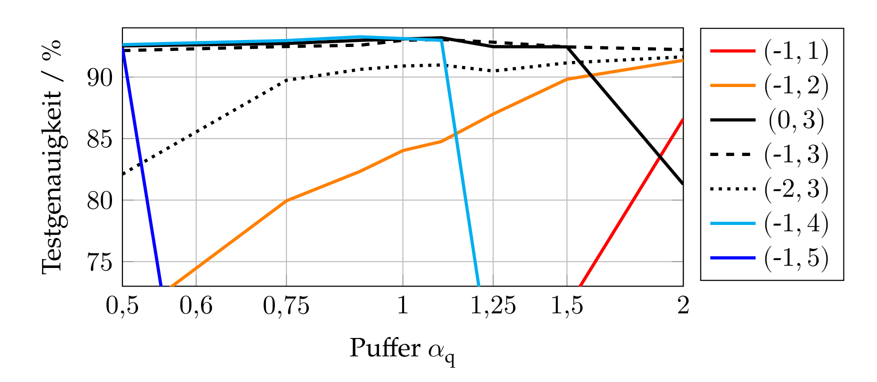


Abbildung 4.10 Einfluss des Puffers α_q auf die Testgenauigkeit bei verschiedenen Verteilungen der acht Quantisierungsbits für VGG-10 mit CIFAR-10. Die Legende gibt die Ganzzahlbits von Gewichten und Aktivierungen als Tupel $(L_{q,I,W}, L_{q,I,A})$ an. Die korrespondierenden CNNs wurden für jeweils 30 Durchläufe trainiert.

Im Folgenden wird der Puffer α_q aus Gl. (3.5) untersucht, welcher für die Schätzung eines geeigneten Skalierungsfaktors für die Aktivierungen verwendet wird. Problematisch bei der Schätzung ist, dass die Aktivierungen je nach Charge $\mathcal{B}$ stark variieren können, da u. U. mangels passendem Eingabebild in der Charge einige Filterkerne $\mathbf{k}_l$ nicht aktiviert bzw. nur schwach aktiviert werden. Hier erweist sich die Verwendung einer großen Chargengröße $N_{\mathcal{B}}$ ebenfalls (vgl. Abschnitt 4.2) als hilfreich, da aufgrund des Gesetzes der großen Zahlen die beschriebene Entartung der Aktivierungen vernachlässigbar ist. Um numerische Probleme auszuschließen, wird der maximale temporäre Skalierungsfaktor $z_{l,\mathcal{B}}$ auf den Wert 12 begrenzt.

Trotz dieser Maßnahmen fällt eine Interpretation von α_q a priori schwierig, da dieser Parameter als zulässiger Wert für einen Überlauf (vgl. [28]) oder als Puffer, der einen Überlauf verhindern soll, eingesetzt werden kann. Abbildung 4.10 deutet darauf hin, dass die optimale Wahl von α_q sowohl von der Bitverteilung der Aktivierungen als auch der Gewichte abhängt. Allgemein deutet $\alpha_q < 1$ auf eine Pufferwirkung hin, während für $\alpha_q > 1$ Überläufe toleriert werden.

In Abb. 4.10 wird die Testgenauigkeit in Abhängigkeit von α_q für verschiedene Bit-Kombinationen dargestellt. Da künstliches Quantisierungs-

rauschen verwendet wird, kommt es nicht zu einer Degradierung der Genauigkeit infolge von Quantisierungseffekten. Im Zentrum steht bei diesem Versuch die Bit-Kombination $L_{\mathrm{q,W}} = (-1, 9)$ und $L_{\mathrm{q,A}} = (3, 5)$. Diese Kombination zeigt eine sehr geringe Abhängigkeit der Testgenauigkeit von α_{q} und weist ein lokales Maximum bei $1{,}1$ auf. Eine Erhöhung der Ganzzahlbitlänge der Aktivierungen (Abb.: $(-1, 4)$, $(-1, 5)$) bzw. der Gewichte (Abb.: $(0, 3)$) bei einer 8-Bit-Quantisierung führen zu einer Degradierung der Genauigkeit bei steigendem α_{q}. Dies ist mit den tendenziell größeren Skalierungsfaktoren bzw. stärkerem Rauschen zu begründen, das die Konvergenzeigenschaften negativ beeinflusst. Je größer die Ganzzahlbitlängen, desto kleiner der Wert von α_{q}, bei dem die Degradierung beginnt. Bit-Kombinationen, die zu kleineren Amplituden der Gewichte und Aktivierungen führen, verhalten sich umgekehrt und führen zu leichten Konvergenzproblemen.

Beide Versuche deuten in Summe auf die Bitverteilung $(-1, 9)$ für Gewichte und $(3, 5)$ für Aktivierungen hin, da diese einerseits im Bereich hoher Genauigkeiten liegt und andererseits robust gegenüber der Wahl von α_{q} ist. Letzteres verspricht ferner eine Robustheit bei breiteren Faltungsschichten. Im Folgenden wird $\alpha_{\mathrm{q}} = 1{,}1$ verwendet, da die Testgenauigkeit über α_{q} bei der verwendeten Bitverteilung hier ein Maximum besitzt. Die beiden Versuche sollen nur qualitativ einen Eindruck für eine gute Wahl der Parameter dieses Quantisierungsverfahrens vermitteln. Für spezifische Anwendungen ist eine Ermittlung mit Validierungsdaten sinnvoll, wobei sich die Werte an den hier ermittelten Werten orientieren können. Die Wahl der Quantisierungsparameter hängt ferner stärker mit der Wahl der Topologie zusammen als mit der des Datensatzes.

Einfluss des Vortrainings

Im Folgenden wird das Vortraining aus Abschnitt 3.4.3 mit zehn Iterationen zur Bestimmung der initialen Skalierungsfaktoren evaluiert. Hierzu werden verschiedene CNN-Konfigurationen jeweils mit und ohne Vorverarbeitung jeweils fünf Durchläufe trainiert. Jedes Experiment wird anschließend 25-mal wiederholt, um eine Konvergenzrate zu berechnen. Hierzu beschreiben r_{90}, r_{75} und r_{20} jeweils den Anteil der CNNs, die nach den fünf Durchläufen eine Testgenauigkeit von mindestens 90%, 75% bzw. 20% erzielen.

Tabelle 4.11 Konvergenzraten r_{90}, r_{75} und r_{20} in Prozent für verschiedene Topologien bei jeweils 25 Wiederholungen.

Vortraining				✓	✓	✓
	r_{20}	r_{75}	r_{90}	r_{20}	r_{75}	r_{90}
VGG-10	100	100	100	100	100	100
VGG-13	100	100	76	100	100	100
VGG-16	96	44	0	100	96	60
VGG-19	16	0	0	100	92	56

In Tab. 4.11 sind die Konvergenzraten für verschiedene Topologien bei einer Verteilung der Quantisierungsbits von $(-1, 9)$ für Gewichte und $(3, 5)$ für Aktivierungen zusammengefasst. Diese Ergebnisse zeigen, dass das Quantisierungsverfahren ohne Vortraining bei tieferen CNNs Konvergenzprobleme hat, die durch ein Vortraining stark reduziert werden. Die bei tieferen CNNs hohen Konvergenzraten r_{75} und r_{90} nach fünf Durchläufen deuten darauf hin, dass die CNNs durch das Vortraining deutlich schneller konvergieren.

In Tab. 4.12 sind die Konvergenzraten für verschiedene ungünstige Bitverteilungen (vgl. Tab. 4.10) und für gröbere Quantisierungen jeweils für VGG-10 gegeben. Für $L_{\mathrm{q,W}} = (-1, 9)$ und $L_{\mathrm{q,A}} = (1, 7)$ ist kein Unterschied bei den Konvergenzraten durch das Vortraining feststellbar. Die Genauigkeit nach fünf Durchläufen liegt in allen insg. 50 Fällen zwischen 83% und 89%. Durch das Vortraining wird allerdings die mittlere Genauigkeit von $85{,}63$ auf $88{,}06\%$ erhöht. Für die anderen beiden 8-Bit-Verteilungen tritt durch das Vortraining eine Verbesserung der Konvergenzraten auf. Während bei einer 6-Bit-Quantisierung (vorletzte Zeile) keine Konvergenzprobleme vorkommen, treten diese bei einer 4-Bit-Quantisierung (letzte Zeile) auf und können durch ein Vortraining abgemildert werden.

Durch das Vortraining des VGG-10-Netz bei einer Verteilung der Quantisierungsbits von $(-1, 9)$ für Gewichte und $(3, 5)$ für Aktivierungen erhöht sich die Testgenauigkeit von $93{,}25\%$ auf $93{,}75\%$.

Insgesamt zeigt das Vortraining alle gewünschten Eigenschaften, d. h. eine Verbesserung der Konvergenzeigenschaften und der Genauigkeit. Daher wird im Folgenden immer ein Vortraining verwendet.

Tabelle 4.12 Konvergenzraten r_{90} und r_{75} in Prozent für verschiedene Bitverteilungen. Die Konvergenzrate r_{20} beträgt in allen Fällen 100%.

Vortraining				✓	✓
$L_{\mathrm{q,W}}$	$L_{\mathrm{q,A}}$	r_{75}	r_{90}	r_{75}	r_{90}
(-1,9)	(1,7)	100	0	100	0
(0,8)	(4,4)	100	76	100	100
(4,4)	(4,4)	88	8	100	100
(-1,7)	(3,3)	100	100	100	100
(-1,5)	(3,1)	96	4	100	12

Einfluss der Feinabstimmung

Im Folgenden werden alle Datensätze aus Abschnitt 4.1 für VGG-10 und VGG-16 jeweils mit und ohne Feinabstimmung untersucht. Die auf drei Versuchen gemittelten Ergebnisse sind in Tab. 4.13 aufgeführt.

Tabelle 4.13 Testgenauigkeit der Datensätze für VGG-10 und VGG-16 vor der Konvertierung (a_{VK}), nach der 8-Bit-Quantisierung der Gewichte (a_{8Bit}) und nach der Feinabstimmung (a_{FA}) jeweils in %. Zusätzlich wird die Standardabweichung des Quantisierungsfehlers σ_{q} nach der Feinabstimmung sowie der Anteil in % der durch die Feinabstimmung geänderten Quantisierungsstufen der Gewichte ($|\mathcal{H}_\Delta|$) angegeben. Die Standardabweichung des Quantisierungsfehlers vor der Feinabstimmung liegt aufgrund der Regularisierung leicht unterhalb des erwarteten Wertes von $2^{-L_{\mathrm{q,F,W}}} \cdot 12^{-0,5} \approx 5{,}6 \cdot 10^{-4}$.

	VGG-10					VGG-16				
	a_{VK}	a_{8Bit}	a_{FA}	$10^4\sigma_{\mathrm{q}}$	$\lvert\mathcal{H}_\Delta\rvert$	a_{VK}	a_{8Bit}	a_{FA}	$10^4\sigma_{\mathrm{q}}$	$\lvert\mathcal{H}_\Delta\rvert$
CIFAR-10	93,75	93,73	93,68	2,75	33,2	94,67	94,67	94,50	1,24	4,5
SVHN	97,24	97,23	97,14	1,96	14,7	97,44	97,38	97,40	0,87	1,5
NEU	98,15	98,27	98,47	1,75	18,6	98,93	98,95	98,42	0,95	2,2
RSDDs	98,07	98,04	97,27	1,39	15,0	98,93	98,90	99,09	1,22	6,1

Die Genauigkeiten von VGG-16 sind für alle Datensätze höher als für VGG-10. Sowohl für VGG-10 als auch für VGG-16 ist tendenziell ein geringer Verlust an Genauigkeit beim Verwenden der Feinabstimmung feststellbar, obwohl diese die geschätzte Standardabweichung des Quantisierungsrauschens reduziert. Bei der Quantisierung auf acht Bits ohne Feinabstimmung entsteht dagegen nahezu kein Verlust. Die unterschiedlichen Genauigkeiten von a_{VK} und a_{8Bit} unterscheiden sich in den meisten

Fällen erst in der zweiten Nachkommastelle, da sich die Hinzugabe von künstlichem Quantisierungsrauschen und die stochastische Rundung der Gewicht infolge der Quantisierung stochastisch identisch verhalten. Weitere Experimente ohne stochastisches Runden führen zu einem ähnlichen Abfallen der Genauigkeit, da hier wie bei der Feinabstimmung den Gewichten deterministisch Werte zugeordnet werden. Das hierdurch indizierte Rauschen folgt zwar der gleichen Gleichverteilung, führt jedoch aufgrund des Histogramms der Gewichte (s. u.) zu anderen Ergebnissen. Obwohl bei VGG-10 ein deutlich höherer Anteil $|\mathcal{H}_\Delta|$ der Gewichte durch Feinabstimmung die Quantisierungsstufe ändert, ist beim VGG-16 durch die Feinabstimmung die Standardabweichung des Quantisierungsrauschens geringer. Dies ist auf den höheren Anteil an Nullen bei VGG-16 zurückzuführen. Insgesamt ergibt sich durch die Quantisierung ein leichtes Abfallen der Genauigkeit im Vergleich zur Verwendung der Standard-CNN-Architekturen aus den Abschnitten 4.3-4.3.2. Diese fällt für größere CNNs geringer aus. Ferner eignet sich die Feinabstimmung nur bei groben Quantisierungen (s. Abschnitt 4.4.2).

Channel-Pruning

Die Ergebnisse für das *Channel-Pruning* der quantisierten CNNs sind in Tab. 4.14 gegeben. Die Ergebnisse sind weitestgehend unabhängig von λ_{CH}. Im Folgenden bezeichnet a_{chp} die Genauigkeit nach Konvertierung und *Channel-Pruning* und a_{8bit} bzw. a_{FA} die Genauigkeiten nach der Quantisierung und ggf. Feinabstimmung für die *Channel-Pruning* verarbeiteten CNNs. Die Genauigkeit a_{VK} vor der Konvertierung stimmt mit der aus Tab. 4.13 überein und ist daher nicht aufgelistet. Ferner wird mit N_{chp} die mittlere gerundete Anzahl an entfernten Filterkernen $\mathbf{k}_l$ und mit r_{Red} der mittlere Anteil in % der entfernten Gewichte an der Gesamtanzahl $|\mathcal{H}|$ bezeichnet. Das gemäß der vorgestellten Verfahren modifizierte VGG-16-Netz besteht aus ca. 14,7 Mio. Parametern mit 4.224 Filterkernen und ein VGG-10-Netz aus ca. 1,74 Mio. Parametern mit 1.152 Filterkernen.

Die erreichte Genauigkeit a_{chp} nach dem *Channel-Pruning* ist in allen Fällen geringer die vor der Konvertierung a_{VK}. Dies ist die direkte Folge aus dem Entfernen von Filterkernen. Bei der stochastischen Quantisierung auf acht Bit ist kein deutlicher Unterschied bei der Genauigkeit

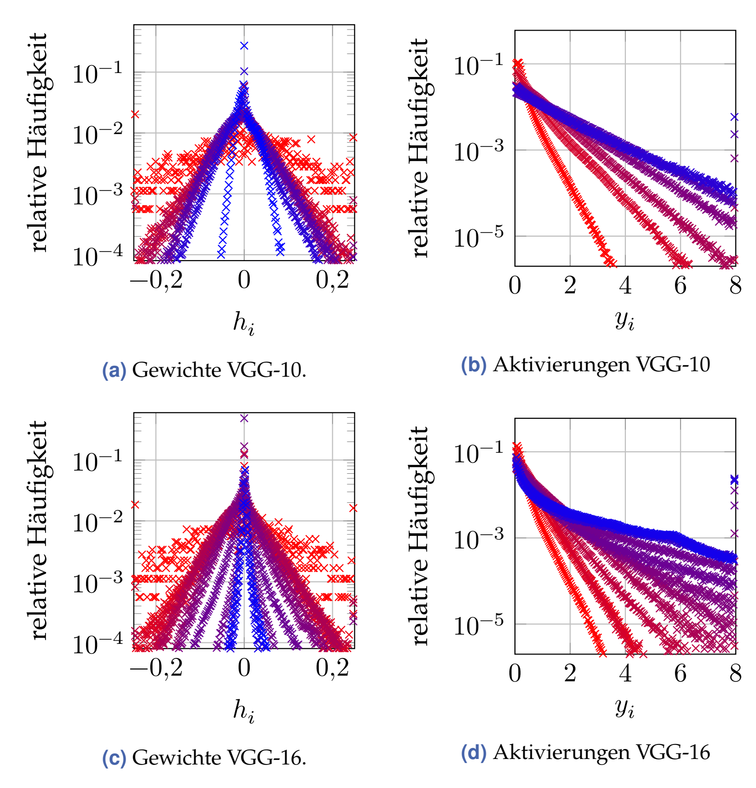
Tabelle 4.14 Testgenauigkeit und Reduktion der Gewichte der CNNS für VGG-10 und VGG-16 bei der Verwendung von *Channel-Pruning*.

	VGG-10					VGG-16				
	a_{chp}	a_{8bit}	a_{FA}	N_{chp}	r_{Red}	a_{chp}	a_{8Bit}	a_{FA}	N_{chp}	r_{Red}
CIFAR-10	93,33	93,32	93,44	16	6,1	94,11	94,07	93,80	1.108	50,4
SVHN	97,12	97,08	97,00	54	10,6	97,06	97,08	97,02	1.925	72,0
NEU	96,92	96,82	96,17	237	62,3	98,78	98,86	98,86	1.696	76,3
RSDDs	97,45	97,64	97,38	325	78,1	98,52	98,57	99,03	2.489	90,9

feststellbar. Auch hier führt die Feinabstimmung aus den oben beschriebenen Gründen zu einer leichten Verschlechterung der Genauigkeit. Erwähnenswert ist hier, dass in allen Konstellationen durch Feinabstimmung rund 30 % der Gewichte verändert werden.

Die Anzahl der entfernten Filterkerne bzw. die Reduktionsraten der Gewichte deuten darauf hin, dass eine komplexere Aufgabe, die sich aus dem Datensatz ergibt, eine höhere Kapazität des CNN erfordert. Dies führt dazu, dass bei kleineren CNNS weniger Filterkerne entfernt werden können. So werden bei VGG-10 mit CIFAR-10 im Mittel 16 Kerne und damit rund 6 % der Gewichte entfernt, während bei VGG-16 mit RSDDs 2.489 Filterkerne bzw. knapp 91 % der Gewichte entfernt werden. Auffallend ist zudem, dass der Anteil der entfernten Filterkerne (für CIFAR-10 1,4 % bei VGG-10 und 58 % bei VGG-16) deutlich geringer ist als der Anteil der entfernten Gewichte. Dies deutet darauf hin, dass vor allem in den hinteren Schichten Filterkerne bzw. Gewichte entfernt werden.

Durch *Channel-Pruning* können bei moderaten Verlusten der Genauigkeit mitunter hohe Reduktionsraten erzielt werden. Die Ergebnisse deuten ferner darauf hin, dass die CNN-Topologie bzw. -Kapazität sich an der Komplexität des Datensatzes orientiert. Insbesondere tiefe CNNs, die in den hinteren Schichten vergleichsweise wenig Filterkerne haben, erscheinen vorteilhaft.



(a) Gewichte VGG-10.

(b) Aktivierungen VGG-10

(c) Gewichte VGG-16.

(d) Aktivierungen VGG-16

Abbildung 4.11 Normierte Histogramme der Gewichte h_i und Aktivierungen y_i aller Faltungsschichten von VGG-10 und VGG-16 mit CIFAR-10. Bei den Aktivierungen wird der Wert Null nicht berücksichtigt. Je rötlicher die Farbe, desto weiter vorne im CNN ist die Schicht, und je bläulicher, desto weiter hinten.

Analyse der Histogramme

Im Folgenden werden die Histogramme der Gewichte und Aktivierungen beispielhaft für den CIFAR-10-Datensatz und die Implikationen für die Topologie untersucht. Abbildung 4.11 zeigt die relativen Häufigkeiten der einzelnen Quantisierungsstufen für die Gewichte und Aktivierungen. Auffallend hierbei ist, dass je weiter vorne eine Schicht ist (rötlicher), desto breiter ist die Verteilung der relativen Häufigkeit der Gewichte und desto steiler sinkt die Verteilung der Aktivierungen ab. Insbesondere in den hinteren Schichten führt dies zu einem sehr hohen Anteil an Nullen

bei den Gewichten (s. bläuliche Ausreißer bei $h_i = 0$ in Abb. 4.11 a und c.). Dies ist mit der höheren Anzahl an Gewichten zu begründen, die bei der VGG-Topologie in hinteren Schichten aufsummiert werden, um eine Aktivierung zu erhalten (vgl. Gl. (2.4)), was aufgrund der Skalierung mit s_l (s. Gl. (3.3)) tendenziell kleinere Gewichte und größere Aktivierungen zur Folge hat.

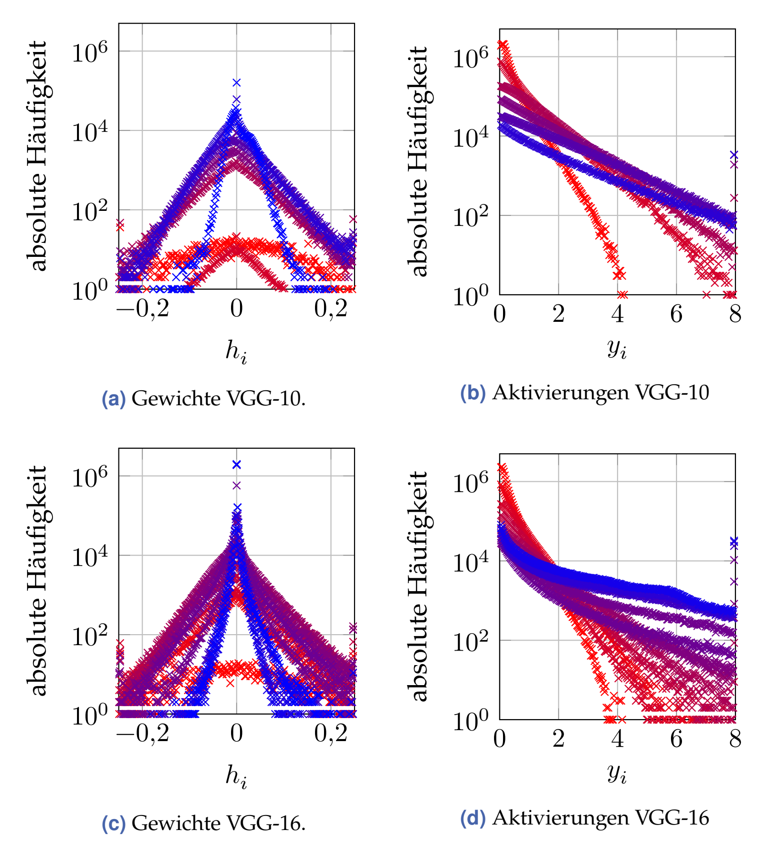


Abbildung 4.12 Histogramme der Gewichte h_i und Aktivierungen y_i aller Faltungsschichten von VGG-10 und VGG-16 mit CIFAR-10. Bei den Aktivierungen wird der Wert null nicht berücksichtigt. Je rötlicher die Farbe, desto weiter vorne im CNN ist die Schicht, und je bläulicher, desto weiter hinten.

Jedoch sinkt die absolute Anzahl an betragsmäßig tendenziell großen Gewichten in den hinteren Schichten kaum (s. Abb. 4.12). Lediglich der Anteil kleiner Gewichte erhöht sich. Dies zeigt, dass der zur Verfügung stehende Wertebereich hinreichend gut ausgenutzt wird und dass eine hohe Redundanz in den hinteren Schichten insbesondere von tieferen CNNs zu erwarten ist[2]. Eine hohe Anzahl an Gewichten hat somit keinen direkten Einfluss auf den Klassifikator. Jedoch wird aufgrund des hohen relativen Fehlers bei der Quantisierung, dem zwingend erforderlichen Vortraining und dem höheren Rechenaufwand die maximale Breite – nicht jedoch die Tiefe – des CNN faktisch durch das Quantisierungsverfahren begrenzt.

Ferner häufen sich die Skalierungsfaktoren für VGG-10 zwischen den Werten $0{,}2$ und $0{,}7$. Diese Häufung tritt bei VGG-16 jedoch nur für Skalierungsfaktoren der ersten sieben Faltungsschichten auf. Hintere Schichten haben 512 Filterkerne und die Werte ihrer Skalierungsfaktoren haben eine breitere Verteilung zwischen 0 und 6. Die Verteilung ist stärker in Richtung 6 ausgeprägt, je weiter hinten die Schicht ist. Dies plausibilisiert die Ergebnisse und die Notwendigkeit des Vortrainings (s. o.). Schichten mit mehr als 512 Filterkernen sind insgesamt für das Quantierungsverfahren potentiell ungeeignet, da deren Gewichte sich eng um den Wert 0 verteilen, was die Quantisierung fehleranfälliger macht.

Bei einer geringen Anzahl an Filterkernen, wie bspw. in der ersten Schicht von VGG-10 mit 32 Filterkernen, sind die Gewichte betragsmäßig tendenziell groß. Hier wird der zur Verfügung stehende Wertebereich der Aktivierungen suboptimal ausgenutzt.

Insgesamt hat die Topologie des CNN einen nicht vernachlässigbaren Einfluss auf die Quantisierung, da die Verteilung der Eingabe $\mathbf{X}$, die Anzahl an Filterkernen L und die Verteilung der Gewichte $\mathbf{H}$ einer Schicht die Verteilung der Aktivierungen $\mathbf{Y}$ derselben beeinflusst. Beide Verteilungen dürfen nicht degenerieren, da dies den Quantisierungsfehler insbesondere bei zu kleinen Werten vergrößert oder zu konvergenzbedingten Schwierigkeiten im Training führt. Der Quantisierungsfehler ist jedoch durch das Hinzufügen von künstlichem Quantisierungsrauschen und dem stochastischen Runden bei der Quantisierung reduzierbar. Dar-

[2] Diese These wird im vorherigen Abschnitt über das *Channel-Pruning* verifiziert, da ein Großteil der Gewichte in hinteren Schichten entfernt werden kann.

über hinaus hat eine degenerierte Verteilung der Aktivierungen einen negativen Einfluss, da sie gleichzeitig als Eingabe für die nächste Schicht dient. Bei einer geeigneten Topologie, d. h. in kleinen bzw. schmalen CNNs, zeigt das Verfahren seine Stärken, da eine adaptive Quantisierung vermieden werden kann und die gesamte Inferenz in der gleichen 8-Bit-Architektur implementiert werden kann.

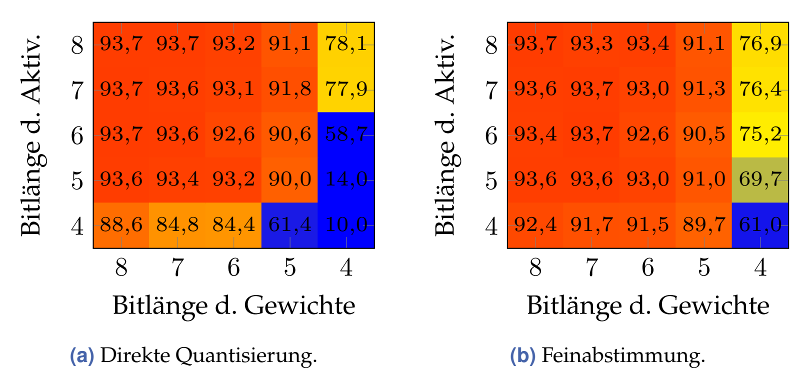
4.4.2 Ergebnisse für gröbere Quantisierung

Im Folgenden wird der Einfluss einer gröberen Quantisierung von VGG-10 und VGG-16 am Beispiel des CIFAR-10 Datensatzes diskutiert. Für jede Kombination der Aktivierungs- und Gewichtsbitlänge wird dafür das entsprechende CNN für 100 Durchläufe trainiert. Es findet ein Vortraining statt und künstliches Quantisierungsrauschen wird während des Trainings hinzugefügt.

Für VGG-10 sind die Ergebnisse in Abb. 4.13 illustriert. Bei der direkten Quantisierung ohne Feinabstimmung tritt eine schwerwiegende Degradierung der Genauigkeit bei weniger als fünf Aktivierungsbits bzw. weniger als sechs Bit für Gewichte auf. Durch die Feinabstimmung wird zwar der Fehler bei der Quantisierung der Gewichte (s. Tab. 4.13) reduziert, jedoch ist der Einfluss auf die Genauigkeit gering. Lediglich bei einer Quantisierung mit vier Aktivierungsbits kann die Genauigkeit erhöht werden. Dies ist darauf zurückzuführen, dass der mittlere quadratische Fehler zwischen der Ausgabe des Teilnetzes und der Inferenz (ohne künstliches Quantisierungsrauschen) minimiert wird, was eine gröbere Quantisierung der Aktivierungen ermöglicht.

Bei VGG-16 ist der Bereich ohne Degradierung der Genauigkeit bei der direkten Quantisierung mit jeweils mindestens sechs Bit deutlich kleiner (s. Abb. 4.14). Durch Feinabstimmung wird hier jedoch der Bereich insbesondere für eine grobe Quantisierung der Gewichte vergrößert. So ist bspw. eine Quantisierung mit vier Bit möglich. Im Gegensatz zu VGG-10 ist keine nennenswerte Verbesserung bei gröberer Quantisierung der Aktivierungen feststellbar.

Feinabstimmung führt insbesondere bei großen CNNs und grober Quantisierung zu Verbesserungen. Bei großen CNN-Topologien ist eine hinreichend große Redundanz im CNN verfügbar [33], die durch das aktive Abbilden durch Feinabstimmung der Gewichte auf diskrete



(a) Direkte Quantisierung.

(b) Feinabstimmung.

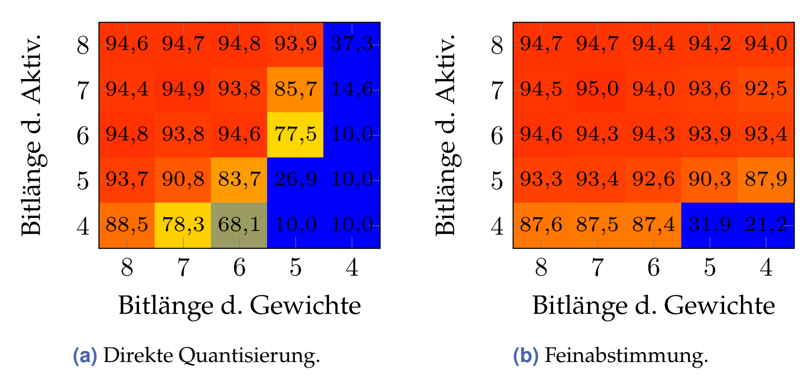
Abbildung 4.13 Erzielte Genauigkeit in % mit und ohne Feinabstimmung abhängig von der Bitlänge für VGG-10.

Quantisierungsstufen reduziert werden kann. Bei kleinen CNNs reicht die Redundanz nicht aus, weswegen mehr Bits erforderlich sind. Tiefere CNNs sind darüber hinaus sensitiver gegenüber Störungen der Aktivierungen. Daher ist hier eine höhere Anzahl an Bits für die Aktivierungen erforderlich.

Der Effekt der Quantisierung auf die Zielfunktion des Lerndatensatzes ist in Abb. 4.15 maßstabsgetreu dargestellt. Für die Visualisierung wird das Verfahren nach [100] mit den CNNs nach der Konvertierung verwendet. Für jede Koordinate, die ausgewertet wird, wird das CNN mit der angegeben Bitlänge quantisiert, welche für Gewichte und Aktivierungen identisch ist. Bei der Auswertung werden ebenfalls die Aktivierungen quantisiert. Somit zeigt die abgebildete Zielfunktion das Verhalten des quantisierten Netzes.

Der Vergleich der drei Zielfunktionen zeigt, dass mit steigender Bitlänge die Zielfunktionen insbesondere im Bereich des Minimums glatter werden. Bei einer geringeren Bitlänge verliert dieser Bereich seine konvexen Eigenschaften zunehmend und wirkt verrauschter. Infolgedessen kommt es zu einem Abfall der Testgenauigkeit, da das Minimum während des Trainings nicht mehr zwangsläufig mit dem Minimum nach der Quantisierung zusammenfällt.

Für eine zu grobe Quantisierung treten darüber hinaus schon während des Trainings Konvergenzschwierigkeiten auf. Dies führen bei vier Bit zu



(a) Direkte Quantisierung.

(b) Feinabstimmung.

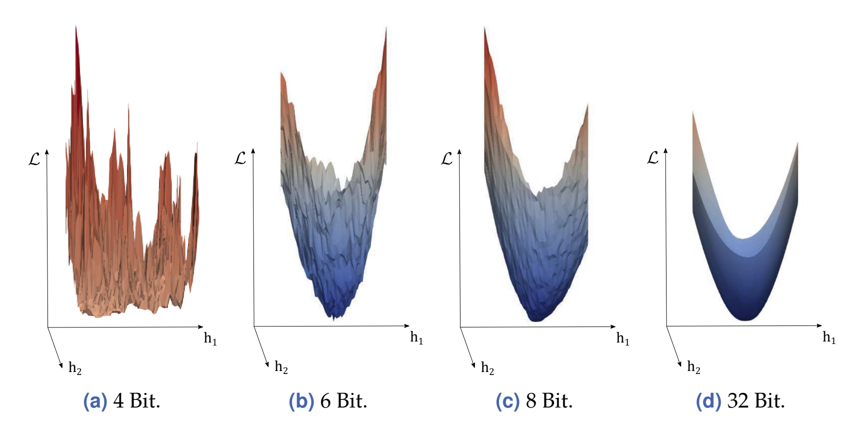
Abbildung 4.14 Erzielte Genauigkeit in % mit und ohne Feinabstimmung abhängig von der Bitlänge für VGG-16.

einem Gebirge ohne eindeutiges Minimum und mit vielen Ausreißern und ähneln den anderen Funktionen aus Abb. 4.15 in keinster Weise. Die Schwierigkeiten sind auf eine zu niedrige Auflösung der Aktivierungen zurückzuführen: Während durch Feinabstimmung der Verlust bei der Quantisierung der Gewichte selbst bei vier Bit ausgeglichen wird, wird der Einfluss der Aktivierungen nur beim Lernen berücksichtigt und kann nicht nachträglich verändert werden. Insbesondere das Abschneiden der Aktivierungen y_i mit $0 < y_i < {}^{\Delta_{\mathrm{q,A}}}\!/_{2}$ verhindert eine Konvergenz beim Training. Infolgedessen führt eine zu kleine Bitlänge der Aktivierungen dazu, dass die Zielfunktion kein markantes Minimum zeigt.

4.4.3 Fazit

In diesem Abschnitt wird das in Abschnitt 3.4 vorgestellte Quantisierungsverfahren für CNNs untersucht. Neben ausführlichen Analysen zu Hyperparametern wird die Bitverteilung für Gewichte und Aktivierungen sowie Erweiterungen des Verfahrens betrachtet. Es zeigt sich, dass jeweils sechs Bit für die Quantisierung von Aktivierungen und Gewichten ausreichend sind. In bestimmten Fällen reichen sogar vier Bit für die Gewichte aus.

Außerdem zeigt sich, dass das Quantisierungsverfahren zwei weitere Einflussfaktoren hat: den Datensatz und die CNN-Topologie. Große bzw. tiefe CNNs haben schlechtere Konvergenzeigenschaften und benötigen



(a) 4 Bit. (b) 6 Bit. (c) 8 Bit. (d) 32 Bit.

Abbildung 4.15 Zielfunktion für verschiedene Quantisierungsbitlängen bei Verwendung eines VGG-16-Netzes mit dem CIFAR-10-Lerndatensatz. Visualisierung nach [100].

daher zwingend ein Vortraining. Jedoch müssen die CNNs groß genug entworfen werden, um eine gewisse Redundanz aufzubauen, die für die Quantisierung ggf. mit Feinabstimmung genutzt werden kann. Bei CIFAR-10 mit VGG-10 ist diese bspw. zu klein für eine Quantisierung der Gewichte mit vier Bit, da das CNN für diesen Datensatz unterdimensioniert ist.

Channel-Pruning reduziert die Redundanz eines CNNs und beschleunigt somit die Inferenz. Das Verfahren ist bei einer zu groben Quantisierung nicht sinnvoll, da die Redundanzen für die Quantisierung benötigt werden. Für ein 8-Bit-CNN führt das Verfahren aber zu einem sinkenden Ressourcenbedarf der Inferenz.

Zu breite CNNs, d. h. CNNs mit einer großen Anzahl bzw. Tiefe von Filterkernen in ihren Schichten, sind für das Quantisierungsverfahren potentiell problematisch, da gemäß Gl. (2.4) eine zu große Tiefe D des Eingabetensors $\mathbf{X}$ bzw. eine zu große Anzahl an Filterkernen L dazu führt, dass mehr Terme aufsummiert werden. Da die maximale Größe der zugehörigen Ausgabekarte $\mathbf{Y}_l$ begrenzt ist, führt dies zu betragsmäßig tendenziell kleineren Gewichte, was zu konvergenzbedingten Schwierigkeiten in der Lernphase führen kann.

Insgesamt eignet sich das Quantisierungsverfahren für konventionelle CNNs und stellt Bedingungen insbesondere an deren Breite. Eine hinsichtlich der Kapazität effiziente Topologie ist darüber hinaus abhängig vom Datensatz, mit dem das entsprechende CNN trainiert wird. Daher ist die datensatzspezifische Suche einer geeigneten Topologie unter Berücksichtigung der Effekte der Quantisierung hilfreich. Ein Algorithmus für den metaheuristischen Entwurf von CNNs wird in Abschnitt 3.5 vorgestellt.

Das Quantisierungsverfahren wurde ferner in Detektions- [184] und Segmentierungsaufgaben [201] erfolgreich verwendet. Während bei der Detektion eine 8-Bit-Quantisierung ausreicht, erfordert die Segmentierung bis zu zwölf Bit für die Gewichte aufgrund der speziellen Netztopologie und der Breite einzelner Schichten des verwendeten U-Net nach [128].

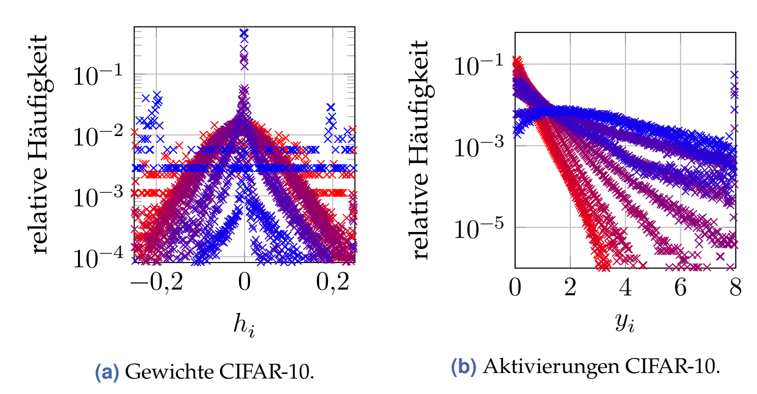
4.5 Evaluation des heuristischen CNN-Entwurfs innerhalb der Verarbeitungskaskade

Im Folgenden wird der evolutionäre Algorithmus aus Abschnitt 3.5 an den Datensätzen aus Abschnitt 4.1 erprobt. Hierfür wird die Heuristik in die Verarbeitungskaskade gemäß Abschnitt 3.6 mit einer 8-Bit-Quantisierung eingebettet, d. h. es wird die datensatzspezifische Vorverarbeitung (aSVST) und Augmentierung (FAARO) verwendet und nach Beendigung der Heuristik findet eine Konvertierung, *Channel Pruning* und eine Quantisierung auf acht Bit statt. Vortraining wird im Initialisierungsschritt der Metaheuristik angewandt. Eine Feinabstimmung wird aufgrund der Ergebnisse für acht Bit aus Abschnitt 4.4.1 nicht durchgeführt.

Das Ziel ist es, eine Topologie für die zu quantisierenden CNNs zu finden, bei der einerseits im Quantisierungsschritt keine Degradierung der Genauigkeit auftritt und die andererseits dem Ressourcenbedarf der einzelnen Datensätze gerecht wird. Das fertige CNN soll auf dem Block-RAM eines FPGAs der `Xilinx7`-Serie implementierbar sein. Da der Speicherplatz für die Gewichte und Aktivierungen hier der limitierende Faktor ist, wird der maximale Speicher der Fitnessfunktion auf rund 95% des verfügbaren Block-RAM eines Vertreters der `Xilinx7`-Serie

gesetzt. Es gilt $s_{\mathrm{MH}}^{\max} = 6.800\,\mathrm{KB}$. Zum Berechnen der Fitness wird jeweils ein Validierungsdatensatz verwendet, der aus $10\,\%$ der ursprünglichen Trainingsdaten besteht. Die restlichen $90\,\%$ werden zum Trainieren der Topologien in der Metaheuristik verwendet.

Die Metaheuristik wird mit $N_{\mathrm{MH},0} = 24$ Topologien initialisiert, wovon aufgrund der Duplizierung von 12 zufälligen Topologien jeweils zwei identisch sind, die nach dem Vortraining für $n_{\mathrm{MH},0} = 6$ Durchläufe trainiert werden. Die Anzahl an Durchläufen steigt während der $T = 10$ Iterationen linear von vier auf acht. Eine höhere Anzahl an Iterationen und Durchläufen führt zu einer längeren Rechendauer, jedoch auch zu einer tendenziell höheren Genauigkeit am Ende. Je nach Datensatz liegt die Rechendauer bei dieser Wahl der Durchläufe und Iterationen bei rund 5 bis 36 h. Nach der letzten Iteration wird das fitteste CNN 50 Durchläufe trainiert. Die jeweilige Anfangslernrate bei einem Training beträgt $2 \cdot 10^{-3}$. Für die Parameter der Fitnessfunktion gilt gemäß empirischer Vorversuche $\lambda_{\mathrm{a}} = \lambda_{\mathrm{H}} = 1$ und $\lambda_{\mathrm{s}} = \lambda_{\mathrm{o}} = 10$.



(a) Gewichte CIFAR-10. **(b)** Aktivierungen CIFAR-10.

Abbildung 4.16 Normierte Histogramme der Gewichte h_i und Aktivierungen y_i aller Faltungsschichten des von der Metaheuristik gefundenen CNNs für CIFAR-10. Bei den Aktivierungen wird der Wert Null nicht berücksichtigt. Je rötlicher die Farbe, desto weiter vorne im CNN ist die Schicht, und je bläulicher, desto weiter hinten.

4.5.1 Ergebnisse

Der zeitliche Verlauf der Populationen $\mathfrak{P}_{\mathrm{MH},t}$ nach jeder Iteration ist in Abb. 4.17 beispielhaft für CIFAR-10 dargestellt. In der Metaheuristik sind

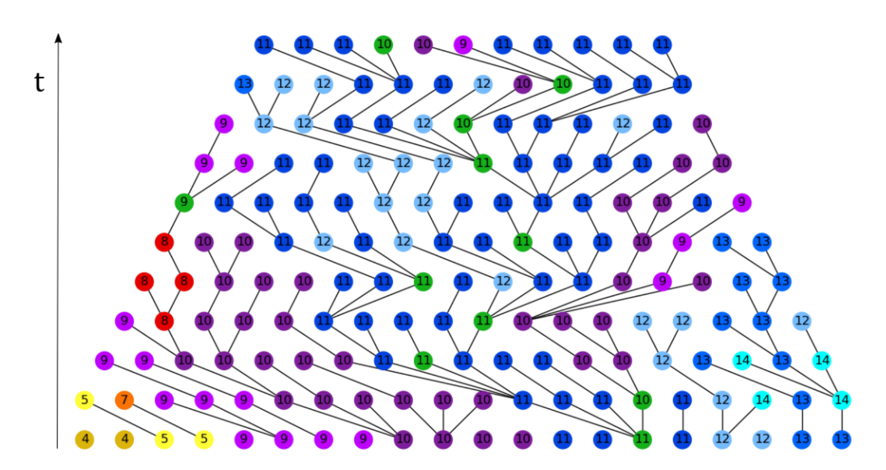


Abbildung 4.17 Entwicklung der Population am Beispiel des CIFAR-10-Datensatzes. Jede Zeile repräsentiert eine Population nach einer vollständigen Iteration, wobei die unterste Zeile die Initialpopulation $\mathfrak{P}_{\mathrm{MH},0}$ zeigt. Die Kanten kennzeichnen die Übergänge der einzelnen Topologien innerhalb einer Iteration. Die Anzahl der Schichten ist farblich codiert. Die fitteste Topologie eines Iterationsschritts ist grün gefärbt. Obwohl nach der Initialisierung eine Vielfalt an verschiedenen Topologien existieren, dominieren bereits nach der dritten Iteration Topologien mit zehn bis zwölf Faltungsschichten.

bereits früh Topologien mit zehn bis zwölf Schichten dominant, während Topologien anderer Länge aus der Population verdrängt werden. Der Verlauf ist für die anderen Datensätze prinzipiell gleich. Lediglich die Populationen konzentrieren sich um andere Werte für die Tiefe der CNNs.

In Abb. 4.16 sind die Histogramme von Aktivierungen und Gewichte der konvertierten CIFAR-10-Topologie dargestellt. Verglichen mit den Histogrammen für VGG-16 in Abb. 4.11 haben die Aktivierungen einen sehr ähnlichen Verlauf, während die Histogramme der Gewichte mitunter eine deutlich breitere Glockenkurve bilden, was die Quantisierung begünstigt. Dies ist einerseits über die geringere Anzahl an Gewichten pro Schicht begründbar und deutet andererseits auf einen Einfluss der Quantisierung auf die Suche nach der Topologie hin. Da die Quantisierungseffekte bereits während des Trainings berücksichtigt werden, erreichen für die Quantisierung ungeeignete Topologien eine schlechtere Validierungsgenauigkeit und damit eine schlechtere Fitness.

Dies zeigt, dass nicht nach einer bestimmten Topologie gesucht wird, sondern Topologien gefunden werden, die prinzipiell ähnliche Eigenschaften bzgl. Klassifikationsgenauigkeit und Ressorcenaufwand haben. Bei der Reproduktion der Ergebnisse wurden für die Datensätze Topologien gefunden, die sich um wenige Schichten und die Verteilung der Filterkerne unterscheiden, aber sehr ähnliche Werte hinsichtlich der Fitness liefern. Weitere Versuche haben gezeigt, dass die Wahl der Fitnessfunktion das Ergebnis stark beeinflussen kann, wodurch eine Abwägung zwischen den beiden Optimierungszielen sinnvoll ist. Der Übersichtlichkeit halber wird im Folgenden für jeden Datensatz nur eine repräsentative Topologie betrachtet.

Die quantitativen Ergebnisse des jeweils besten CNN, das durch die Metaheuristik ermittelt wurde, sind für die vier Datensätze in Tab. 4.15 gegeben. Die erzielten Genauigkeiten sind mit Ausnahme von CIFAR-10 mit denen von VGG-16 aus Tab. 4.14 vergleichbar. Jedoch ist der Ressourcenbedarf für die Implementierung von VGG-16 deutlich höher, da für Gewichte 14.375 KB und für Aktivierungen 201 KB (SVHN, CIFAR-10) bzw. 804 KB (NEU, RSDDs) an Speicher benötigt werden. Ferner werden bei VGG-16 $35{,}85$ Mio. (SVHN, CIFAR-10) bzw. $143{,}4$ Mio. (NEU, RSDDs) MAC-Operationen pro Bild ausgeführt.

Tabelle 4.15 Eigenschaften der gefundenen Topologien. Die Genauigkeiten sind in Prozent angegeben und ähneln denen aus Tab. 4.14. Die Anzahl der MAC-Operationen (in Mio.) sowie der Speicher für Gewichte (in KB) und Aktivierungen (in KB) bezieht sich auf das Modell nach dem *Channel-Pruning*. Die Laufzeit ist in hh:mm angegeben. Mit $L^{(i)}$ wird die Anzahl an gefundenen Filterkernen der i-ten Schicht bezeichnet.

	CIFAR-10	SVHN	NEU	RSDDs
a_{VK}	92,91	97,28	98,91	98,56
a_{chp}	92,24	96,88	98,37	98,51
a_{8bit}	92,49	96,86	98,37	98,49
Fitness	40,91	466,8	7.554	1.123
Laufzeit	24:12	36:23	5:22	22:23
Tiefe L_{MH}	10	10	9	12
MAC-Ops	14,25	12,67	25,91	18,31
Aktiv.	184	156	382	330
Gewichte	2.283	1.303	781	1.016
$L^{(1)}$	102	85	32	39
$L^{(2)}$	27	47	50	28
$L^{(3)}$	106	81	45	54
$L^{(4)}$	123	118	94	32
$L^{(5)}$	55	125	98	71
$L^{(6)}$	169	133	91	99
$L^{(7)}$	248	197	127	138
$L^{(8)}$	238	179	182	118
$L^{(9)}$	219	129	152	82
$L^{(10)}$	325	112		313
$L^{(11)}$				82
$L^{(12)}$				131

Die gefundenen Topologien sind tendenziell tief und schmal, was die Quantisierung begünstigt (vgl. Abschnitt 4.4.1). Mit neun bis zwölf Faltungsschichten bewegt sich die Tiefe im Bereich von VGG-11 und VGG-16. Die Breite, d. h. die Anzahl der Filterkerne pro Schicht übersteigt im Gegensatz zu VGG-Topologien selten den Wert 250. Ferner ist in einigen Topologien ein Alternieren zwischen breiten und dünnen Schichten zu beobachten. Dies führt sowohl in der dicken als auch in der dünnen Schicht zu einer Reduktion der MAC-Operationen, da entweder die Tiefe des Eingabetensors D oder die Tiefe des Ausgabetensors L gering sind

(vgl. Abschnitt 2.2.2). Gemäß der Interpretation nach [2] neigen solche Topologien zu einem geringeren Grad an Auswendiglernen und einer besseren Fähigkeit zu generalisieren. Die Bestrafung größerer Netze führt daher nicht nur zu einem CNN, das auf FPGAs implementierbar ist, sondern hat auch eine geringere Neigung zur Überanpassung.

Des Weiteren ist festzustellen, dass komplexere Datensätze zu einem größeren Ressourcenbedarf tendieren, was sich mit der Beobachtung aus Abschnitt 4.3.2 deckt. Der tatsächliche Ressourcenbedarf, den die Metaheuristik ermittelt, hängt darüber hinaus mit der erzielbaren bzw. erzielten Genauigkeit ab: Je höher die Genauigkeit und damit die Fitness, desto geringer fällt ein hoher Strafterm ins Gewicht, insbesondere aufgrund der hyperbolischen Abhängigkeit von Fitness bzgl. der Genauigkeit. Dies führt dazu, dass die Topologie bei CIFAR-10 stärker hinsichtlich des Ressourcenbedarfs als bzgl. der Genauigkeit optimiert wird. Umgekehrt ergibt sich aufgrund der hohen Genauigkeiten bei RSDDs eine große Topologie. Bestätigen lässt sich diese Beobachtung durch die Betrachtung der Reduktionsrate der Gewichte r_{Red} durch *Channel-Pruning* des besten CNN der Endpopulation. Diese liegt für CIFAR-10 bei $6{,}4\,\%$ und für RSDDs bei $63{,}8\,\%$[3]. Der hohe Ressourcenbedarf bei RSDDs ist auch auf mögliche Fehler bei der Annotierung zurückzuführen (vgl. Abschnitt 4.1), aufgrund derer ein höheres Abstraktionslevel erforderlich ist, da zusätzlich zur eigentlichen Aufgabe *echte* Fehler von *unechten* Fehlern unterschieden werden müssen.

4.5.2 Fazit

Die Metaheuristik findet abhängig vom Datensatz Topologien, deren Eigenschaften sich mit den Erwartungen decken, da komplexere Datensätze mehr CNN-Kapazitäten benötigen und tiefe und schmale CNNs bevorzugt werden. Letztere haben nicht nur einen geringeren Speicherbedarf für die Gewichte, sondern neigen weniger zum Auswendiglernen des Lerndatensatzes (vgl. [2]), was sich ebenfalls positiv auf den Speicherbedarf auswirkt. Da die Quantisierungseffekte bereits in der Lernphase berücksichtigt werden, findet die Metaheuristik ferner eine Topologie,

[3] Für SVHN liegt diese bei $20{,}7\,\%$ und bei NEU bei $16{,}6\,\%$, was auf eine im Vergleich zu den Versuchen aus Abschnitt 4.4 geringe Redundanz hinweist

die sich gut für die Quantisierung eignet, da ungeeignete Topologien die Fitness negativ beeinflussen.

Die Wahl der Hyperparameter λ_a, λ_H, λ_s und λ_o sowie die Anzahl an Durchläufen beeinflussen das Ergebnis mitunter stark. Beispielsweise erhöht sich die Genauigkeit für CIFAR-10, wenn der Strafterm um den Faktor zehn kleiner ist und die Anzahl an Durchläufen verdoppelt wird, auf 93,29 %, wobei das resultierende CNN breiter wird und eine höherer Reduktionsrate von ca. 28 % durch das *Channel-Pruning* aufweist. Das gefundene CNN ähnelt damit dem quantisierten VGG-10-Netz nach *Channel-Pruning* hinsichtlich Anzahl der Gewichte, welche 1.59 KB entspricht, und der Genauigkeit von 93,33 %. Gleichzeitig verdoppelt sich in diesem Versuch die Laufzeit der Metaheuristik für CIFAR-10 von rund 24 h auf 47 h. Die Anzahl der Durchläufe pro Iteration ist an die Komplexität bzw. die Konvergenzgeschwindigkeit anzupassen, da der degradierende Einfluss einer evolutionären Operation nach dem anschließenden Training zumindest ausgeglichen werden muss, sodass eine Degradierung nur infolge einer schlechteren Topologie stattfindet. Zusammenfassend ist für die Hyperparameter eine datensatzspezifische Wahl erforderlich, wenn der Datensatz zu komplex ist. Ferner führt eine hohe Validierungsgenauigkeit auch dazu, dass potentiell höhere Strafterme tolerierbar sind. Dies führt bei Datensätzen mit einer hohen tatsächlich erreichten Genauigkeit zu tendenziell größeren CNN-Topologien.

Insgesamt eignet sich die Metaheuristik insbesondere für industrielle Datensätze, da diese weniger komplex sind und eine hohe Genauigkeit erreicht werden kann. Hierbei ist eine korrekte Annotierung essentiell. Bei komplexeren Datensätzen muss in Einzelfall zwischen Genauigkeit und Ressourcenbedarf abgewägt werden.

5 CNNs für robotergestützte Inspektion

Im zweiten Teil dieser Arbeit wird der Einsatz von CNNs für die robotergestützte Inspektion am Beispiel des *Remanufacturing* untersucht. Dem *Remanufacturing* wird eine in Zukunft zunehmende Bedeutung in der Produktion vorausgesagt, da aufgrund einer möglichen Verknappung wichtiger Grundstoffe wie Kupfer, Silber oder Stahl steigende Rohstoffpreise an den Terminmärkten prognostiziert werden. Dies kann zur Unwirtschaftlichkeit der Produktion in bestimmten Bereichen führen. Das *Remanufacturing* zielt darauf ab, Altprodukte wieder zu neuwertigen Produkten ohne einem Verlust des Gebrauchswerts wiederaufzubereiten, was die Wertschöpfung auf hohem Niveau belässt [161]. Hierfür ist gemäß [108] eine Reihe von Arbeitsschritten erforderlich, die die Demontage, die Inspektion, die Reinigung, die Wiederaufbereitung, die Remontage und eine abschließende Prüfung umfassen.

Dieser Teil dieser Arbeit ist im Rahmen des Projekts *AgiProbot*, das durch die Carl-Zeiss-Stiftung gefördert wird, entstanden. Das Ziel des Projektes ist der Aufbau einer autonomen, lernenden Gesamtfabrik, die das *Remanufacturing* am Beispiel von Elektromotoren mit unbekannter Bauform und unbekanntem Zustand durchführt. Die Gesamtfabrik ist gemäß der fluiden Automatisierung nach [170] entworfen. Diese Arbeit beschäftigt sich mit der Erkennung und Inspektion von Ankern aus Elektromotoren an der sog. *Befundungsstation* [183]. Hierfür wird in Abschnitt 5.1 ein CNN entworfen, welches Anker unterschiedlichster Formen erkennen und die relevanten Komponenten segmentieren kann. Das Segmentierungsergebnis wird anschließend durch klassische Bildverarbeitung verbessert. In Abschnitt 5.2 wird ein bildbasierter Regler vorgestellt, der auf Grundlage von Merkmalen des Segmentierungsergebnisses die Position zwischen einer Kamera und einem Anker findet, welche für eine anschließende Inspektion geeignet ist.

5.1 Robuste Objektsegmentierung mit neuronalen Netzen

Der Inhalt der Abschnitte 5.1.2 bis 5.1.4 wurde größtenteils in [186] veröffentlicht. Abschnitt 5.1.5 stammt ursprünglich aus [187]. Im Folgenden wird der Entwurf eines neuronalen Netzes zur Segmentierung von Komponenten von Altprodukten am Beispiel von Ankern aus Elektromotoren beschrieben. Die Segmentierung des Altproduktes stellt den ersten entscheidenden Schritt für das *Remanufacturing* dar, da die funktionsrelevanten Komponenten des Ankers erkannt und lokalisiert werden. Dies erleichtert die anschließende optische Inspektion. Es wird eine hohe Anforderung an die Genauigkeit der Segmentierung gestellt, da dies die Grundlage für eine bildbasierte Regelung (s. Abschnitt 5.2) und eine anschließende Inspektion ist. Um Anker unabhängig von ihrer Bauform, ihrer Lage oder ihres Produktzustandes robust segmentieren zu können, ist die Verwendung geeigneter Lerndaten sowie das Einbringen von Zusatz- bzw. Expertenwissen maßgebend. Daher wird in Abschnitt 5.1.2 der verwendete Datensatz ausführlich beschrieben, in Abschnitt 5.1.3 die Augmentierung diskutiert, in Abschnitt 5.1.4 Erweiterungen des neuronalen Netzes beschrieben und in Abschnitt 5.1.5 Wissen über den Anker zu einem Modell aggregiert. Das verwendete neuronale Netz basiert auf dem U-Net nach [128].

5.1.1 Stand der Technik

In früheren Arbeiten haben sich neuronale Netze [92] als vorteilhaft für komplexe Bildverarbeitungsaufgaben wie bspw. Klassifikation und Segmentierung herausgestellt.

Der naive Ansatz, neuronale Netze für Segmentierungsaufgaben zu verwenden, ist mehrere Faltungsschichten aneinanderzureihen, da tiefere Netze bessere Eigenschaften hinsichtlich der Merkmalsextraktion aufweisen. Dieser Ansatz ist in praktischen Anwendungen meist nicht sinnvoll, da er eine hohe Rechenintensität aufweist. Das in [5] beschriebene *Segnet* ist eine hinsichtlich Rechen- und Speichereffizienz optimierte Form eines solchen Netzes und weist eine *Autoencoder*-Struktur auf: Durch Heruntertasten wird die Höhe und Breite der Aktivierungskarten zunächst

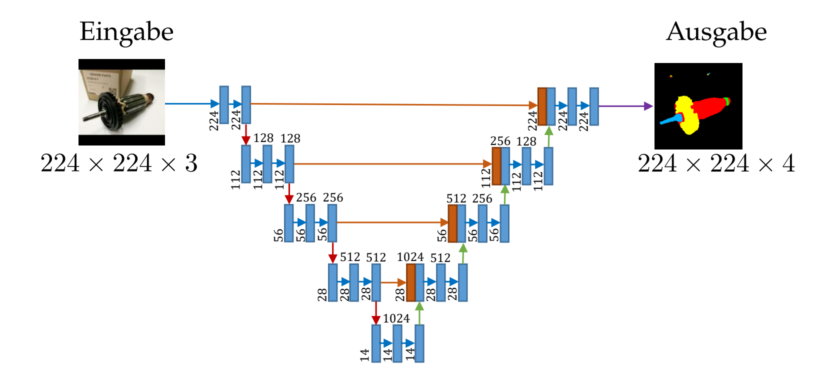


Abbildung 5.1 Schematischer Aufbau des U-Net. Blaue Pfeile beschreiben eine (3×3)-Faltungsschicht mit BN und ReLU-Aktivierung, violette Pfeile eine (1×1)-Faltungsschicht, rote Pfeile eine *Max-Pooling*-Schicht mit $(\cdot)_{\downarrow 2,\mathrm{MP}}$ und grüne Pfeile eine Hochtastschicht $(\cdot)_{\uparrow 2}$. Die orangenen Pfeile sind Querverbindungen, bei denen die Aktivierungen kopiert werden und in einer späteren Schicht an den Aktivierungstensor konkateniert werden. Die Zahlen an den Aktivierungstensoren (blaue Rechtecke) beschreiben ihre Tiefe (waagerecht) und Höhe bzw. Breite (senkrecht).

verkleinert, um anschließend durch Hochtastschichten die ursprüngliche Bildgröße zu erhalten. Da hier das Netzwerk nur auf Effizienz optimiert wird, wird das Potential neuronaler Netze für Segmentierungsaufgaben nicht vollumfänglich genutzt.

Daher hat sich für die semantische Segmentierung das U-Net nach [128] durchgesetzt (s. Abb. 5.1). Das U-Net ist gekennzeichnet durch eine *Autoencoder*-Struktur mit Querverbindungen, die hochgetastete höherwertige Merkmale mit den Merkmalen aus früheren Schichten kombinieren, und erlaubt so ein adaptives Lernen einer geeigneten Tiefe der Struktur. Ferner enthält das U-Net keine vollständig verbundenen Schichten, was das Verwenden nahezu beliebig großer Bilder bei gleicher Netztopologie ermöglicht. Als Aktivierungsfunktion der letzten Schicht wird eine *softmax*-Funktion verwendet. In [128] wird die gewichtete kategorische Kreuzentropie als Zielfunktion verwendet. Das U-Net wurde ursprünglich für die Segmentierung von Zellen in Bildern eines Hellfeldmikroskops verwendet.

Insbesondere bei allgemeineren Segmentierungsaufgaben nehmen die Zielobjekte bzw. Zielobjekte bestimmter Klassen im Bild nur einen ge-

ringen Teil ein, was eine Unausgewogenheit der Klassen im Datensatz bedingt. Dies führt zu schlechteren Ergebnissen. In [153] wird vorgeschlagen, dass jede Klasse einen eigenen Beitrag zur Gesamtzielfunktion leistet, der entsprechend gewichtet wird. Ferner wird gezeigt, dass der generalisierte Sörensen-Dice-Koeffizient (gSDK) nach [29] von den untersuchten Zielfunktionen hierfür am besten geeignet ist.

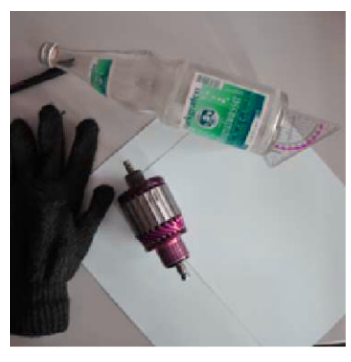
Weitere Erweiterungen betreffen die Architektur. Hierbei werden residuelle Verbindungen zwischen den Faltungsschichten eingebaut [36, 178], Faltungsschichten innerhalb der Querverbindungen eingefügt [179] oder Punktwolken segmentiert [24], was jedoch wiederum die Recheneffizienz verschlechtert.

5.1.2 Datensatz

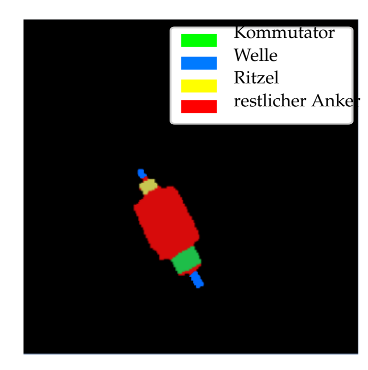
Für das Lernen eines neuronalen Netzes ist eine Vielzahl an annotierten Bildern in einem geeigneten Kontext erforderlich. Im Zuge dieser Arbeit wurde ein relativ kleiner Datensatz von insg. 96 Bildern angefertigt, da für die semantische Segmentierung von Ankerkomponenten aus Elektromotoren kein öffentlich zugänglicher Datensatz auffindbar ist. Der Datensatz ist aufgrund des Kosten- und Zeitaufwandes der Annotierung klein gehalten und wird im Folgenden beschrieben.

Für den Datensatz wurden zum einen eigene Aufnahmen eines Ankers aus einem Anlassermotor der Firma `Bosch` verwendet und zum anderen frei zugängliche Bilder von Ankern von Online-Versandhäusern. Während erstere mehrere irrelevante Objekte im Hintergrund zeigen, besitzen letztere mehrheitlich einen einfarbigen Hintergrund. Für die Verwendung als Eingabe in ein neuronales Netz werden die Bilder zu einem Quadrat zurecht geschnitten und anschließend mittels eines geeigneten Antialiasing-Filter auf 224×224 Pixel herauf- bzw. heruntergetastet. 75 der 96 Bilder werden für das Training verwendet. Die restlichen Bilder sind Testdaten.

Den Ankerkomponenten *Kommutator* (K), *Welle* (W) und *Ritzel* (R) kommt eine besondere Relevanz für das *Remanufacturing* zu, da deren Leitfähigkeit und mechanischen Eigenschaften starken Einfluss auf die Funktionalität des gesamten Elektromotors haben. Damit ist die Segmentierungsaufgabe in zwei Teile zerlegbar. Einerseits wird zwischen Anker (A) und Hintergrund unterschieden und andererseits werden die Bild-



(a) Originalbild $\mathbf{I}$



(b) Grundwahrheit $\mathbf{M}$

Abbildung 5.2 Bei der Annotierung werden basierend auf dem Originalbild für den Anker und seine Komponenten (Kommutator, Welle und Ritzel) Masken erstellt. Bild nach [186].

elemente innerhalb der Ankermaske bzgl. ihrer Klassenzugehörigkeit differenziert. Dies unterscheidet die vorliegende Aufgabe vom Stand der Technik (s. Abschnitt 5.1.1). Daher erhält jedes Pixel die Information, ob es sich innerhalb der Ankermaske befindet und ggf. zu welcher Klasse es gehört. Damit ergibt sich innerhalb der Ankermaske ein Vier-Klassen-Problem aus den drei relevanten Komponenten und dem restlichen Anker. Mit dieser Formulierung wird sichergestellt, dass die Anker bei der Segmentierung in zusammenhängenden Gebieten liegen, und es wird dem Netz implizit vermittelt, dass die Komponenten immer innerhalb der Ankermaske liegen. Die Annotierung ist in Abb. 5.2 als Falschfarbenbild dargestellt. Im Folgenden wird das Eingabebild als $\mathbf{I} \in \mathbb{R}^{(224,224,3)}$, die Grundwahrheit als $\mathbf{M} = [\mathbf{M}_\mathrm{A}, \mathbf{M}_\mathrm{K}, \mathbf{M}_\mathrm{W}, \mathbf{M}_\mathrm{R}] \in \mathbb{R}^{(224,224,4)}$ und das segmentierte Ausgabebild als $\mathbf{P} = [\mathbf{P}_\mathrm{A}, \mathbf{P}_\mathrm{K}, \mathbf{P}_\mathrm{W}, \mathbf{P}_\mathrm{R}] \in \mathbb{R}^{(224,224,4)}$ bezeichnet.

5.1.3 Augmentierung

Aufgrund der geringen Anzahl an Bilddaten (s. Abschnitt 5.1.2) und der vergleichsweise großen Anzahl an Parametern des U-Net von ca. 31 Mio. sind geeignete Methoden zum Lernen invarianter Merkmale unumgänglich. Da sich die Lerndaten mitunter deutlich von den Manifestationen

des Ankers bei der bildbasierten Regelung unterscheiden, eignen sich die Verfahren aus Abschnitt 3.2 und 3.3 nicht. Daher wird im Folgenden Expertenwissen verwendet, um eine geeignete Augmentierungsstrategie zu finden, die in Abschnitt 6.1 evaluiert wird.

Da die Bilder des Lerndatensatzes nur auf Grundlage hoher Diversität der Anker ausgewählt wurden, ist eine geeignete Augmentierung aus den Schwächen des Lerndatensatzes und den Erwartungen an die Bilder der bildbasierten Regelung ableitbar. Diese sind:

- Die Lernbilder verfügen teilweise nur über einfarbige Hintergründe, wodurch das U-Net dazu neigt, Abweichungen vom Hintergrund als Objekt zu erkennen (s. Abschnitt 5.1.2).
- Die Anker des Lerndatensatzes haben eine beliebige, unbekannte Größe, was das Erlernen einer Skaleninvarianz aufgrund der Schwierigkeiten, die es speziell im maschinellen Lernen dabei gibt [89, 106, 146], erschwert.
- Der Anker bei der bildbasierten Regelung kann in unterschiedlichsten Positionen und Perspektiven auftreten.
- Bewegungsunschärfe, schlechte Fokussierung oder die *Antialiasing*-Filterung bei der Abtastratenanpassung bewirken eine Tiefpassfilterung des Eingangsbildes in das U-Net.
- Aufgrund von Beleuchtung und ungünstiger Kamerasensorkalibrierung können Schwankungen von Helligkeit, Kontrast, Sättigung und Farbwerten sowie Rauschen auftreten.

Die Augmentierungsoperationen werden als Kaskade wie in [30] ausgeführt und orientieren sich an den oben genannten Gegebenheiten. Die Kaskadenelemente sind jeweils eine Gruppe von Bildverarbeitungsoperationen mit stochastischer Parametrisierung gemäß Abschnitt 3.3.2. In der ersten Stufe der Kaskade werden Spiegelungen stochastisch zu 50% an der x- und 50% an der y-Achse auf das Bild angewandt. Anschließend wird eine affine Transformation ausgeführt, die aus einer Rotation um einen zufälligen Winkel zwischen $-\frac{\pi}{2}$ und $+\frac{\pi}{2}$, einer Translation, einer Scherung und einer Skalierung besteht. In der dritten Stufe wird *Cutout* nach [35] verwendet, um Verdeckungen zu simulieren. Die vierte Stufe

entspricht der sog. *Hintergrundaugmentierung*. In der letzten Stufe wird zufällig eine Gaußfilterung, eine Bildverschärfung, gaußsches Rauschen, Impulsrauschen, eine Farbwertverschiebung, eine Helligkeitsänderung, eine Sättigungssänderung oder eine Kontraständerung auf das Bild angewandt, um eine unzureichende Segmentierung infolge von schlechter Bildqualität zu vermeiden [37].

Einige Operationen werden im Folgenden näher betrachtet. Eine zentrale Rolle kommt der Skalierung zu, da sie die Größe des Ankers im Bild bestimmt, für die das U-Net zuverlässig arbeitet. Da durch die Ankermaske $\mathbf{M}_{\mathrm{A}}$ aus der Grundwahrheit die Größe s_{A} und Position $(u_{\mathrm{A}}, v_{\mathrm{A}})$ des Ankers im Bild $\mathbf{I}(u, v)$ während der Lernphase bekannt sind, wird die Skalierung gezielt dazu verwendet, um die Objektgröße in einer bestimmten Schwankungsbreite zu halten. Der Bereich, gegenüber den ein neuronales Netz skaleninvariant ist, kann nicht beliebig groß sein, da einerseits eine minimale Größe eines Objektes abhängig vom Schwierigkeitsgrad der Aufgabe erforderlich ist [14, 159, 181] und andererseits die maximale Größe eines Objektes durch das rezeptive Feld des neuronalen Netzes gedeckelt wird [109]. Daher wird der Schwankungsbereich der Objektgröße in dieser Arbeit auf 5 bis 20% der Objektgröße bezogen auf die Bildgröße festgesetzt. Dies entspricht bei einem Bild der Größe 224×224 ca. 2.500 bis 10.000 Pixeln. Um das Bild anzupassen, wird ein Skalierungsfaktor für die Seitenlänge des Bildes berechnet, für den der Anker im Bild die gewünschte Größe hat. Eingedenk der Position des Objektes wird das Bild zurechtgeschnitten oder mit Nullen aufgefüllt und anschließend wieder auf 224×224 Pixel skaliert.

Die Hintergrundaugmentierug fußt auf einer Veröffentlichung von Tremblay et al. [163], die auf dem Konzept der Domänenrandomisierung zum Erlernen der Variabilität realer Daten beruht. Dabei wird das neuronale Netz durch eine zufällige, nicht zwangsweise realistische Umgebung dazu gezwungen, wesentliche Merkmale des Objekts zu erlernen. Für die Zwecke dieser Arbeit wird die Verwendung der Hintergrundaugmentierung auch aufgrund der einfarbigen Hintergründe motiviert. Hierzu wird aus der Grundwahrheit des Bildes $\mathbf{I}$ die Ankermaske $\mathbf{M}_{\mathrm{A}}$ extrahiert. Diese wird verwendet, um den Anker auszuschneiden und vor ein zufälliges Hintergrundbild $\mathbf{I}_{\mathrm{H},i}$ aus dem `dtd`-Datensatz [25] zu legen, welcher verschiedenste Texturen zeigt. Ein ähnliches Verfahren

wird auch in [7] verwendet. Das Ergebnis wird mit einen Tiefpassfilter $\mathbf{k}_{\text{Gauss}}$ gefaltet, um Artefakte beim Ausschneiden zu vermeiden. Mit der elementweisen Multiplikation $\odot$ ergibt sich das resultierende Bild $\mathbf{I}_{\text{H}}$ zu

$$\mathbf{I}_{\text{H}} = \mathbf{k}_{\text{Gauss}} ** \left(\mathbf{M}_{\text{A}} \odot \mathbf{I} + (1 - \mathbf{M}_{\text{A}}) \odot \mathbf{I}_{\text{H},i}\right). \tag{5.1}$$

5.1.4 Erweiterungen

Im U-Net nach [128] wird als letzte Aktivierung eine *softmax*-Aktivierung verwendet, welche für das in Abschnitt 5.1.2 beschriebene Problem ungeeignet ist, da das Ergebnis zu logischen Widersprüchen führt bzw. dazu, dass das Auftreten des Ankers und seiner Komponenten als zusammenhängendes Gebiet nicht sichergestellt ist. Daher werden im Folgenden alternative Aktivierungsfunktionen diskutiert.

Zwar kann eine sigmoide Aktivierung $\sigma_{\text{sig}}(\cdot)$ dazu führen, dass sich die Komponentenklassen nicht gegenseitig ausschließen oder einzelne Komponenten nicht als Teil des Ankers erkannt werden, aber diese Aktivierung lässt die Schätzung unabhängiger Masken zu. In einem Nachverarbeitungsschritt können die genannten Widersprüche aufgelöst werden. Mit der Ausgabe des U-Net vor der letzten Aktivierung $\mathbf{Y} = [\mathbf{Y}_{\text{A}}, \mathbf{Y}_{\text{K}}, \mathbf{Y}_{\text{W}}, \mathbf{Y}_{\text{R}}]$ werden die geschätzten Masken $\mathbf{P}$ zu $\mathbf{P} = \text{f}_{\text{bin}}(\sigma_{\text{sig}}(\mathbf{Y}))$, wobei $\text{f}_{\text{bin}}(\cdot)$ das Ergebnis binarisiert.

Im Folgenden werden zwei weitere Aktivierungsfunktionen vorgestellt, die die logischen Widersprüche sukzessive beim Training auflösen, was jedoch zu schlechteren Konvergenzeigenschaften führt. Im ersten, dem multiplikativen Ansatz mit Sigmoid-Funktion (MSig), wird ausgeschlossen, dass Komponenten außerhalb der geschätzten Ankermaske liegen. Die geschätzten Masken $\mathbf{P}$ berechnen sich gemäß

$$\begin{aligned}
&\mathbf{P}_{\text{A}} = \text{f}_{\text{bin}}(\sigma_{\text{sig}}(\mathbf{Y}_{\text{A}})) \text{ und} \\
&\mathbf{P}_{x} = \mathbf{P}_{\text{A}} \cdot \text{f}_{\text{bin}}(\sigma_{\text{sig}}(\mathbf{Y}_{x})) \text{ mit } x \in \{\text{K}, \text{W}, \text{R}\}.
\end{aligned} \tag{5.2}$$

Beim zweiten Ansatz wird durch eine *softmax*-Funktion $\sigma_{\text{SM}}(\cdot)$ das gegenseitige Ausschließen der Ankerklassen sichergestellt. Die sog. multiplikative Softmax-Aktivierung (MSM) ergibt sich zu

$$\begin{aligned}
&\mathbf{P}_{\text{A}} = \text{f}_{\text{bin}}(\sigma_{\text{sig}}(\mathbf{Y}_{\text{A}})) \text{ und} \\
&[\mathbf{P}_{\text{K}}, \mathbf{P}_{\text{W}}, \mathbf{P}_{\text{R}}, \mathbf{P}_{\text{X}}] = \mathbf{P}_{\text{A}} \cdot \text{f}_{\text{bin}}(\sigma_{\text{SM}}([\mathbf{Y}_{\text{K}}, \mathbf{Y}_{\text{W}}, \mathbf{Y}_{\text{R}}, \mathbf{0}])).
\end{aligned} \tag{5.3}$$

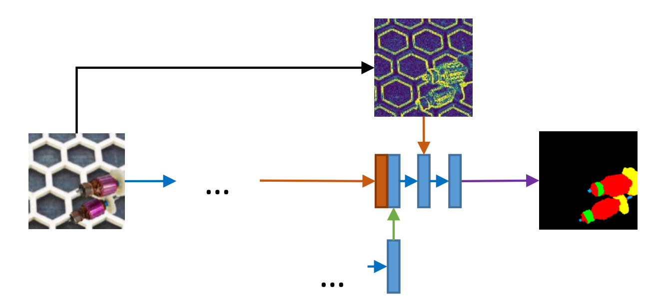


Abbildung 5.3 Hinzugabe von Kanteninformationen in das U-Net. Bild nach [186].

Hierbei wird eine Platzhalterklasse X verwendet für den Fall eines Pixels innerhalb der Ankermaske ohne Zugehörigkeit zu einer der drei Komponenten.

Da die semantischen Objektgrenzen des Ankers auf Kanten im Bild fallen, kann die Hinzunahme von Kanteninformation in einen der letzten Aktivierungstensoren des U-Net (s. Abb. 5.1) zu einer verbesserten Segmentierung führen. In dieser Arbeit wird der Marr-Hildreth-Operator (engl. *Laplacian of Gaussian*) [112] zur Kantenextraktion auf das zum Grauwertbild umgewandelte Eingangsbild angewandt. Anschließend wird der Betrag des Ergebnisses mit $0{,}02$ gewichtet, beim Wert 1 abgeschnitten und als weitere Eingabe in das U-Net verwendet. Es wird untersucht, ob eine Konkatenierung oder Addition vor der letzten (3×3)-Faltungsschicht zu einer Verbesserung führt. Die Änderung des U-Net ist in Abb. 5.3 dargestellt.

5.1.5 Modellbasierte Objekterkennung

Im Folgenden wird ein modellbasiertes Verfahren beschrieben, das das Segmentierungsergebnis stabilisiert und Fehler teilweise korrigieren kann. Das Verfahren eignet sich für Bildsequenzen desselben Ankers wie bspw. Videos oder bei der bildbasierten Regelung und beruht auf folgenden Annahmen:

- In der Aufnahme des Ankers sind immer der Kommutator und entweder ein oder beide Enden der Welle sichtbar und werden vom U-Net korrekt segmentiert.
- Die Anker in aufeinander folgenden Bildern unterscheiden sich nur geringfügig in ihrer Perspektive und ihre Masken können über eine euklidische Transformation in Deckung gebracht werden.
- Fehler bei der Segmentierung treten aufgrund kleiner, ungünstiger Transformationen (vgl. [4]) auf, gegen die das CNN variant ist, und kommen gegenüber einer korrekten Segmentierung selten vor.

Die erste Annahme wird zum Verringern von Fehldetektionen verwendet, da zusammenhängende Objekte im segmentierten Bild für die Erkennung als Anker über mindestens zwei zusammenhängende Gebiete verfügen müssen, wovon eines als Kommutator und das andere als Welle erkannt wird. Zusammenhängende Objekte, denen eines dieser Gebiete fehlt, sind als Fehldetektionen interpretierbar. Um einzelne Pixel, die einer der beiden Klassen zugeordnet werden, nicht als zusammenhängendes Gebiet zu interpretieren, wird gefordert, dass ein solches Gebiet über mind. 10 Pixel verfügt. Da ein Ritzel am Anker je nach Bauform optional bzw. oft mit geringem Aufwand abbaubar ist, ist das Erkennen als Anker nicht abhängig von dessen Vorhandensein. Falls zu viele Kommutatoren und Wellen bzw. mehr als ein Ritzel erkannt wird, werden die kleinsten erkannten Gebiete entfernt. Durch eine ungünstige Perspektive ist es möglich, dass kein Kommutator erkannt wird und das Objekt fälschlicherweise als Fehldetektion zurückgewiesen wird. Dieser Fehler tritt insbesondere dann auf, wenn die optische Achse der Kamera und die Rotationsachse des Ankers kollinear sind. Um diesen Fehler bei der bildbasierten Regelung zu vermeiden, muss entweder sichergestellt werden, dass sich die Kamera während der Regelung zu keinem Zeitpunkt an einer solchen Position befindet oder dass der Regler selbstständig die Situation auflösen kann.

Auf Grundlage der beiden anderen Annahmen wird das Modell des Ankers zur Stabilisierung der U-Net-Ausgabe aufgebaut. Der in der dritten Annahme beschriebene Effekt, der durch Augmentierung zwar reduziert, aber nicht komplett verhindert wird, kann bei der Schätzung der Lageparameter für den bildbasierten Regler zu schwerwiegenden

Fehlern führen, die eine erfolgreiche Regelung unmöglich machen. Daher werden die Ausgaben vorheriger Bilder zu Hilfe genommen, um den Fehler einzudämmen. Das Verfahren wird im Folgenden beschrieben.

Für die Stabilisierung ist ein Ensemble von Modellen des Ankers erforderlich. Ein einzelnes Modell ist eine Repräsentation eines Ankers aus einer früheren Eingabe in das U-Net. Ein Modell besteht aus dem Bild des Ankers ohne Hintergrund $\mathbf{I}_{\mathrm{M},i} = \mathbf{I} \odot \mathbf{P}_{\mathrm{A}}$, den extrahierten ORB-Merkmalen (s. u.) und einer Konfidenz $\mathbf{C}_{\mathrm{M},i}$, die zu Beginn mit $\mathbf{C}_{\mathrm{M},i} = 0{,}75 \cdot \mathbf{P}$ initialisiert wird. Für die Stabilisierung der aktuellen Eingabe wird das passende Modell des Ensembles dazu verwendet, um die aktuelle Schätzung der Ankermaske mit der Ankermaske des Modells zu fusionieren. Gleichzeitig wird mit der aktuellen Schätzung die Konfidenz des Modells aktualisiert, aus der sich nach einer Binarisierung die Ankermaske des Modells ergibt.

Falls das Ensemble leer ist oder dem Anker aus dem Eingabebild kein Modell zugeordnet werden kann, wird der Anker in der Eingabe als neues Modell hinzugefügt. Das setzt voraus, dass ein Anker aus dem Eingabebild $\mathbf{I}$ extrahiert werden kann (s. o.).

Nach der Segmentierung werden innerhalb der geschätzten Ankermaske $\mathbf{P}_{\mathrm{A}}$ im Eingabebild $\mathbf{I}$ ORB-Merkmale nach [130] detektiert. Die gefundenen Merkmale werden über die euklidische Distanz mit den Merkmalen der bereits vorhandenen Modelle abgeglichen. Punkte aus der Eingabe und dem jeweiligen Modell mit ähnlichen Merkmalen bilden Paare. Falls genügend Punktepaare vorhanden sind, kann eine euklidische Transformationsmatrix $\hat{\mathbf{T}}_{\mathrm{M},i}$ zwischen der Eingabe und dem i-ten Modell geschätzt werden. Die affine Transformation wird auf das Eingabebild und das dazugehörige Segmentierungsergebnis angewandt. Diese werden zu $\mathbf{I}^{(i)}$ und $\mathbf{P}^{(i)}$. Anschließend wird die Ähnlichkeit zwischen Eingabe und Modell i berechnet, falls der Jaccard-Koeffizienten zwischen den Ankermasken von Eingabe und Modell den Wert $0{,}7$ überschreitet. Die Ähnlichkeit entspricht dann dem Korrelationskoeffizienten der Pixel innerhalb der Ankermaske $\mathbf{P}^{(i)}$ zwischen den Bildern der Anker. Falls die maximale Ähnlichkeit größer als $0{,}9$ ist, wird die dazugehörige Konfidenz aktualisiert. Andernfalls wird der Anker der Eingabe als ein neues Modell hinzugefügt (s. o.). Dieses Verfahren ist in Abb. 5.4 dargestellt.

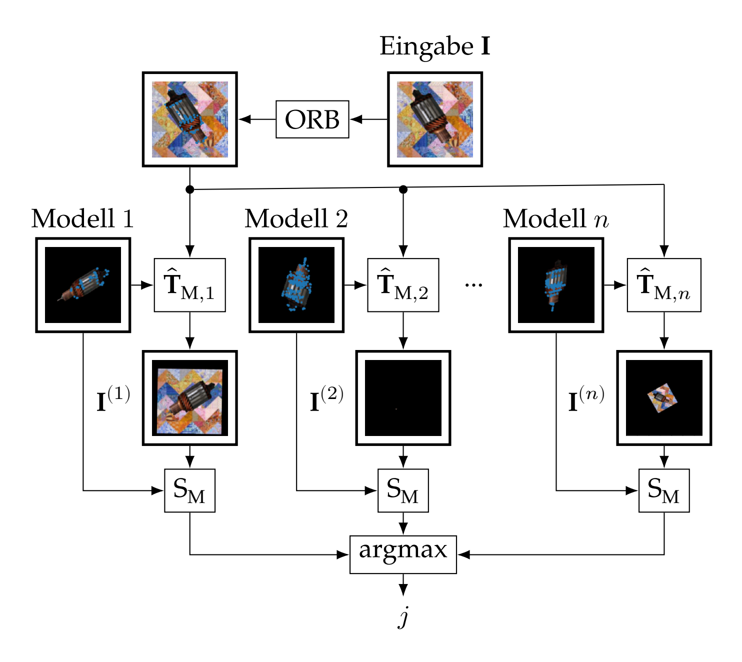


Abbildung 5.4 Schematischer Ablauf der Zuordnung von Eingabebild $\mathbf{I}$ zum jeweiligen Modell $j = 1, \dots, n$. Mit $\mathrm{S_M}(\cdot, \cdot)$ wird die Funktion zur Berechnung der Ähnlichkeit bezeichnet.

Die Aktualisierung der Konfidenzen $\mathbf{C}_{\mathrm{M},j}$ des Modells mit der höchsten Ähnlichkeit $j = \operatorname{argmax}_i \mathrm{S_M}(\mathbf{I}^{(i)}, \mathbf{I}_{\mathrm{M},i})$ ergibt sich aus einer gleitenden Mittelwertbildung zwischen $\mathbf{C}_{\mathrm{M},j}$ und den geschätzten Masken $\mathbf{P}$ der Eingabe. Es gilt $\mathbf{C}_{\mathrm{A,M},j} \leftarrow \alpha_{\mathrm{M}} \mathbf{C}_{\mathrm{A,M},j} + (1 - \alpha_{\mathrm{M}}) \cdot \mathbf{P}_{\mathrm{A}}$. Die Masken der Komponenten des Modells $\mathbf{P}_{\mathrm{M},j}$ ergeben sich durch die Binarisierung von $\mathbf{C}_{\mathrm{M},j}$. Überschüssige Komponenten werden wie oben beschrieben entfernt. Falls zu wenige Komponenten einer Klasse gefunden werden, wird der Schwellenwert bei der Binarisierung dieser Klasse angepasst. Die stabilisierte Ausgabe des Eingangsbildes $\mathbf{I}$ ergibt sich aus der affinen Transformation von $\mathbf{P}_{\mathrm{M},j}$ mit $\hat{\mathbf{T}}_{\mathrm{M},i}^{-1}$, d. h. die affine Rücktransformation des am besten passenden Modells wird als Schätzung der Masken des aktuellen Bildes verwendet.

5.2 Bildbasierte Regelung mit neuronalen Netzen

Der im Folgenden beschriebene Regler basiert auf [187]. Die bildbasierte Regelung mit Hilfe von neuronalen Netzen hat gegenüber der klassischen bildbasierten Regelung einige Vorteile: Während bei der klassischen bildbasierten Regelung einfache, stabile Merkmale aus der Aufnahme der beweglichen Kamera extrahiert und in eine Bewegung der Kamera überführt werden, erlauben neuronale Netze, die erforderliche Bewegung direkt aus der Eingabe zu schätzen. Aufgrund der hohen Anzahl trainierbarer Parameter und der Komplexität verhalten sich CNNs wie *Black-Box*-Modelle, was die Erklärbarkeit für eine bestimmte Bewegung – insbesondere dann, wenn sie fehlerhaft ist – enorm erschwert. In Abschnitt 5.2.1 wird der Stand der Technik für den Einsatz von CNNs bei der bildbasierten Regelung beschrieben.

In dieser Arbeit wird ein bildbasierter Regler für Anker vorgestellt, der die Vorteile von klassischer bildbasierter Regelung und Regelung mithilfe neuronaler Netze vereint. Dazu wird die stabile Segmentierung, die in Abschnitt 5.1 beschrieben wird, verwendet. Diese erlaubt die stabile Extraktion komplexer Merkmale in Bildern. Dies wird zusammen mit der dazugehörigen Jacobi-Matrix in Abschnitt 5.2.2 beschrieben. Das dazugehörige Regelsystem wird in Abschnitt 5.2.3 thematisiert. Zum Abschluss wird in Abschnitt 5.2.4 auf die Problematik der Skalenvarianz von neuronalen Netzen eingegangen und ein Lösungskonzept hierfür vorgestellt. Das Ziel der Regelung ist, die Kamera so zu positionieren, dass die optische Achse senkrecht zur Rotationsachse des Ankers steht. Aufgrund der annähernden Rotationssymmetrie bilden die gesuchten Punkte einen Kreis um den Anker. Die Zielpositionierung ist in Abb. 5.5 dargestellt.

5.2.1 Stand der Technik

Der Einsatz von neuronalen Netzen für die bildbasierte Regelung unterscheidet sich von der in Abschnitt 2.3.2 beschriebenen klassischen bildbasierten Regelung darin, dass neuronale Netze auf Grundlage eines Eingangsbildes $\mathbf{I}$ die Bildmerkmale $\mathbf{m}$ oder die Steuerungsvariablen

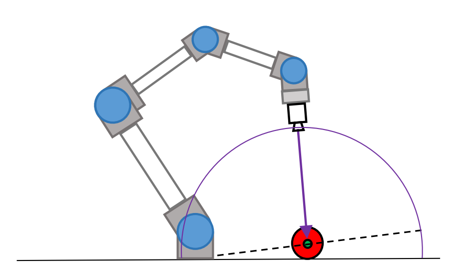


Abbildung 5.5 Zielpositionierung der Kamera zum Objekt. Der violette Pfeil deutet die optische Achse der Kamera an, die orthogonal zur Oberfläche des Ankers steht. Der violette Kreisbogen markiert die gültigen Zielpositionen des optischen Zentrums der Kamera.

direkt schätzen. Ein oft verwendeter Ansatz beruht auf dem Verstärkungslernen (engl. *reinforcement learning*) der Steuerung der beweglichen Kamera [95, 116, 134]. Nachteilig an dieser Methodik ist, dass das Verhalten von neuronalen Netzen nicht vollständig verstanden ist [88, 118], dass neuronale Netze sich mit wenig Aufwand täuschen lassen [53, 73, 119] und dass neuronale Netze auf neuartige Umgebungen unerwartet reagieren können [26].

Die folgenden Verfahren verkleinern die problematischen Auswirkungen des *Black-Box*-Modells, indem dem CNN Zusatzinformationen gegeben werden, nur die Bildmerkmale geschätzt werden oder die Eigenschaften durch perspektivische Verzerrung direkt trainiert werden.

Saxena et al. [135] verwenden ein CNN, um eine Drohne zu steuern. Dabei wird als Eingabe für das Netz das aktuelle Bild der Kamera der Drohne und das gewünschte Bild an der Zielposition übergeben. Das neuronale Netz berechnet dann mithilfe der erlernten Merkmale einen Bewegungsvektor, mit dem das neue Bild der Drohne dem gewünschten Bild ähnlicher ist. Ein ähnliches Verfahren wird von Tokuda et al. in [160] vorgestellt. Jedoch schätzt hier das neuronale Netz die Differenz zwischen aktueller und gewünschter Lage eines Endeffektors aus verschiedenen Eingabebildern. In [6] wird dagegen ein neuronales Netz verwendet, um die Lage von perspektivisch verzerrten Bildern zu schätzen, anhand derer ein Regler entworfen wird, der die theoretische Kameralage für ein

unverzerrtes Bild schätzt. PUANG et al. [124] nutzen ein U-Net [128], um Punktmerkmale zu bestimmen, mit denen eine klassische bildbasierte Regelung erfolgt.

5.2.2 Bildmerkmale und Jacobi-Matrix

Für die Bestimmung der Bildmerkmale für die bildbasierte Regelung wird die geschätzte Ankermaske $\mathbf{P}_\mathrm{A}$ verwendet. Aus der Menge $\mathcal{P}_\mathrm{A} = \{(u,v) \mid \mathbf{P}_\mathrm{A}(u,v) = 1\}_{(u,v)}$ der Bildkoordinatentupel, die bei der Segmentierung dem Anker zugeordnet werden, kann die Position $[u_\mathrm{A}, v_\mathrm{A}]^\mathrm{T}$ des Ankers im Bild durch eine Mittelwertbildung geschätzt werden. Die Position eignet sich als Punktmerkmal (vgl. Abschnitt 2.3.2) für den Regler. Mittels der Koordinatentransformationen aus Gl. (2.8) und (2.9) ergeben sich die finalen Bildmerkmale der Position $[x^\mathrm{B}_{1,\mathrm{A}}, y^\mathrm{B}_{1,\mathrm{A}}]^\mathrm{T}$. Die Zielwerte $[x^{\mathrm{B}*}_{1,\mathrm{A}}, y^{\mathrm{B}*}_{1,\mathrm{A}}]^\mathrm{T}$ werden auf null gesetzt, sodass im Optimum das Zentrum des Ankers in der Bildmitte liegt. Die zugehörige Jacobi-Matrix $\mathbf{J}_\mathrm{x}$ ist in Gl. (2.16) gegeben.

Die Orientierung ergibt sich ebenfalls aus $\mathcal{P}_\mathrm{A}$. Hierfür wird der Eigenvektor mit dem größten Eigenwert aus der Kovarianzmatrix bzgl. der Punktmenge $\mathcal{P}_\mathrm{A}$ berechnet. Die Orientierung $\phi_\mathrm{A} \in [-\pi/2, +\pi/2]$ des Ankers ergibt sich aus der Steigung dieses Eigenvektors. Der Steigungswinkel ϕ_A und der Mittelpunkt des Ankers $[x^\mathrm{B}_{1,\mathrm{A}}, y^\mathrm{B}_{1,\mathrm{A}}]^\mathrm{T}$ definieren eine Gerade. Diese kann gemäß [173] durch den Abstand zum Ursprung ρ_A und den Winkel θ_A der zur Orientierung orthogonalen Gerade parametrisiert werden. Aus der zeitlichen Ableitung der Polarkoordinaten der normalisierten Bildebene [18] und den Geradenparametern ρ_A und θ_A ergibt sich die Jacobi-Matrix für den Winkel θ_A zu

$$\dot{\theta}_\mathrm{A} = \begin{bmatrix} J_{\theta_\mathrm{A}\mathrm{vx}} & J_{\theta_\mathrm{A}\mathrm{vy}} & J_{\theta_\mathrm{A}\mathrm{vz}} & \rho_\mathrm{A}\cos\theta_\mathrm{A} & \rho_\mathrm{A}\sin\theta_\mathrm{A} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}^\mathrm{K} \\ \boldsymbol{\omega}^\mathrm{K} \end{bmatrix}. \quad (5.4)$$

Gemäß [17] ist die Verwendung der Jacobi-Matrix im Optimum eine legitime Approximation der Jacobi-Matrix zum Zeitpunkt t. Bei einer optimalen Kameraposition ist die Ausbreitung der Gerade in die Tiefe vernachlässigbar. Ferner gilt im Optimum $\rho_\mathrm{A} = 0$ aufgrund der Zielposition $[x^{\mathrm{B}*}_{1,\mathrm{A}}, y^{\mathrm{B}*}_{1,\mathrm{A}}]^\mathrm{T}$ (s. o.). Die Jacobi-Matrix aus Gl. (5.4) kann somit zu $\mathbf{J}_\theta = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ vereinfacht werden. Der Winkel θ_A kann

durch einfache geometrische Überlegungen aus der Geradensteigung und dem Ankermittelpunkt bestimmt werden. Der Zielwinkel ist 0, was dazu führt, dass der Anker waagerecht im Bild liegt.

Der Abstand zwischen der Kamera und dem Objekt bestimmt dessen Größe im Bild. Gemäß Abschnitt 5.1.3 kann das U-Net nur Objekte einer definierten Größe segmentieren. Je größer aber das Objekt im Bild ist, desto detailreicher kann der Anker aufgenommen werden. Jedoch muss technisch bedingt die Kamera einen Mindestabstand vom Objekt haben. Es wird angenommen, dass die Objektgröße $s_\mathrm{A} = h_{1,\mathrm{A}} \cdot b_{1,\mathrm{A}} \cdot \xi_\mathrm{A}$ durch die Fläche eines begrenzenden Rechtecks mit den Seitenlängen $h_{1,\mathrm{A}}$ und $b_{1,\mathrm{A}}$ korrigiert um den Faktor ξ_A beschrieben werden kann und dass diese nur vom Abstand Z^K zum Objekt abhängt. Die Änderung der Fläche Δs_A des Ankers abhängig von $h_{1,\mathrm{A}}$ und $b_{1,\mathrm{A}}$ ist dann gegeben durch

$$\Delta s_\mathrm{A} = \xi_\mathrm{A}(h_{1,\mathrm{A}}\Delta b_{1,\mathrm{A}} + b_{1,\mathrm{A}}\Delta h_{1,\mathrm{A}} + \Delta h_{1,\mathrm{A}}\Delta b_{1,\mathrm{A}}). \tag{5.5}$$

Mit den Zusammenhängen

$$\Delta h_{1,\mathrm{A}} = -\frac{h_{1,\mathrm{A}}\Delta Z^\mathrm{K}}{Z^\mathrm{K} + \Delta Z^\mathrm{K}} \text{ und } \Delta b_{1,\mathrm{A}} = -\frac{b_{1,\mathrm{A}}\Delta Z^\mathrm{K}}{Z^\mathrm{K} + \Delta Z^\mathrm{K}} \tag{5.6}$$

und der Abschätzung $(\Delta Z^\mathrm{K})^2 \ll \Delta Z^\mathrm{K} \ll Z^\mathrm{K}$ ergibt sich

$$\begin{aligned} \Delta s_\mathrm{A} &= -2\frac{\xi_\mathrm{A} b_{1,\mathrm{A}} h_{1,\mathrm{A}}}{Z^\mathrm{K}} \cdot \Delta Z^\mathrm{K} \text{ bzw.} \\ \dot{s}_\mathrm{A} &\approx -2\frac{\xi_\mathrm{A} b_{1,\mathrm{A}} h_{1,\mathrm{A}}}{Z^\mathrm{K}} \cdot \dot{Z}^\mathrm{K} =: J_\mathrm{svz} \cdot v_\mathrm{z}^\mathrm{K}. \end{aligned} \tag{5.7}$$

Die korrespondierende Jacobi-Matrix ist dann $\mathbf{J}_\mathrm{s} = \begin{bmatrix} 0 & 0 & J_\mathrm{svz} & 0 & 0 & 0 \end{bmatrix}$. Um die Höhe $h_{1,\mathrm{A}}$ und Breite $b_{1,\mathrm{A}}$ berechnen zu können, wird ein begrenzendes Rechteck in der Ankermaske $\mathbf{P}_\mathrm{A}$ geschätzt, deren Sensorkoordinaten in Koordinaten der imaginären Bildebene transformiert werden. Die aktuelle Größe s_A ergibt sich als Quotient zwischen der Anzahl an Bildpixeln und der Mächtigkeit von $\mathcal{P}_\mathrm{A}$. Der Korrekturfaktor ξ_A wird ebenfalls zu jedem Zeitschritt geschätzt und ergibt sich als Quotient zwischen der Anzahl der Pixel im begrenzenden Rechteck und der Mächtigkeit $|\mathcal{P}_\mathrm{A}|$.

Im Folgenden werden zwei konkurrierende Bildmerkmale für die Orientierung der Kamera entlang der X^K- und Y^K-Achse vorgestellt. Für

das erste Merkmal werden die langen Kanten des Ankers verwendet, welche mit der probabilistischen Hough-Transformation [86] und einem Schwellenwertvergleich der Länge der Kanten berechenbar sind. Bei einer Neigung zwischen Objekt- und Bildebene verlaufen diese Linien nicht parallel, sondern schneiden sich im perspektivischen Zentrum des Bildes. Die optimale Steigung dieser langen Kanten ist identisch mit dem Orientierungswinkel ϕ_A. Kanten, die eine zu große Differenz hierzu aufweisen, werden nicht weiter berücksichtigt. Da sich die Rotationskomponente um die z-Achse gegenüberliegender Kanten ausgleicht, wird der korrespondierende Anteil in der Jacobi-Matrix zu null gesetzt, um Fehler, die entstehen, falls mehrere lange Kanten geschätzt werden, zu vermeiden. Da i. A. eine unbekannte Anzahl N an Kanten gefunden wird, ergibt sich die Jacobi-Matrix zu (vgl. Gl. (5.4))

$$\mathbf{J}_\mathrm{t} = \begin{bmatrix} 0 & 0 & 0 & \rho_{\mathrm{A},0}\cos\psi_{\mathrm{A},0} & \rho_{\mathrm{A},0}\sin\psi_\mathrm{A}, 0 & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & \rho_{\mathrm{A},N}\cos\psi_{\mathrm{A},N} & \rho_{\mathrm{A},N}\sin\psi_{\mathrm{A},N} & 0 \end{bmatrix}. \tag{5.8}$$

Die Alternative hierzu stellt die Bestimmung der Orientierung mithilfe der Schätzung der Bodenebene dar, die durch einen Stütz- und zwei Spannvektoren darstellbar ist. Bei der gesuchten Orientierung der Kamera steht deren optische Achse orthogonale zur Bodenebene. Unter der Annahme, dass die Spannvektoren achsenparallel verlaufen, ist lediglich die Änderung der Tiefe Z^K in X^K- und Y^k-Richtung für den Regler zu bestimmen. Im Gegensatz zur ersten Methode, bei der die optimale Zielposition auf einem Kreis(-bogen) liegt, ist die Zielposition hier eindeutig. Für eine korrekte bildbasierte Regelung ist es erforderlich, dass die Symmetrieachse des Ankers parallel zur Bodenebene ist. Unter der Annahme, dass in einem Großteil des Bildes die Ebene sichtbar ist, können der Stützvektor und die beiden Spannvektoren über eine lineare Regression entlang der x_1^B- und entlang der y_1^B-Koordinate des Tiefenbildes mit einem *Least-Square*-Schätzer mit RANSAC [45] geschätzt werden. Es ergeben sich die beiden Steigungen der Ebene $\nu_\mathrm{x} = {}^{\Delta Z^\mathrm{K}}/_{\Delta X^\mathrm{K}}$ und $\nu_\mathrm{y} = {}^{\Delta Z^\mathrm{K}}/_{\Delta Y^\mathrm{K}}$. Nach einer zeitlichen Differenzierung und Anwenden von Gl. (2.11) ergibt sich

$$\dot{\nu}_\mathrm{x} = -\omega_\mathrm{x}(1+\nu_\mathrm{x}^2) \text{ und } \dot{\nu}_\mathrm{y} = -\omega_\mathrm{y}(1+\nu_\mathrm{y}^2). \tag{5.9}$$

Die zugehörige Jacobi-Matrix $\mathbf{J}_\mathrm{t}$ hat somit lediglich einen Eintrag bei ω_x und ω_y. Die vollständige Bild-Jacobi-Matrix gemäß Gl. (2.10) ergibt sich als Komposition der Jacobi-Matrizen der einzelnen Merkmale. Die schematische Form dieser Matrix ist

$$\begin{aligned} \mathbf{J} &= \begin{bmatrix} \mathbf{J}_\mathrm{x}^\mathrm{T} & \mathbf{J}_\mathrm{s}^\mathrm{T} & \mathbf{J}_\theta^\mathrm{T} & \mathbf{J}_\mathrm{t}^\mathrm{T} \end{bmatrix}^\mathrm{T} \\ &= \begin{bmatrix} \mathbf{J}_\mathrm{xvx} & \mathbf{J}_\mathrm{xvz} & \mathbf{J}_\mathrm{x\omega x} & \mathbf{J}_\mathrm{x\omega z} \\ 0 & J_\mathrm{svz} & 0 & 0 \\ 0 & 0 & 0 & J_{\theta\omega\mathrm{z}} \\ \mathbf{0} & \mathbf{0} & \mathbf{J}_{\psi\omega\mathrm{x}} & \mathbf{0} \end{bmatrix}. \end{aligned} \tag{5.10}$$

5.2.3 Regelsystem

Da die Jacobi-Matrix aufgrund der getroffenen Annahmen dünn besetzt ist und die Merkmale unterschiedlich sensitiv sind, wird ein entkoppelnder Ansatz nach [173] für die bildbasierte Regelung verwendet. Die Struktur der Jacobi-Matrix in Gl. (5.10) erlaubt es, Bewegungen der Kamera in X^k- bzw. Y^k-Richtung, die nicht durch das Punktmerkmal $[x_{1,\mathrm{A}}^\mathrm{B}, y_{1,\mathrm{A}}^\mathrm{B}]^\mathrm{T}$ induziert werden, auszugleichen, und den optimalen Schritt in der X^k- bzw. Y^k-Richtung separat zu berechnen. Beispielsweise führt eine Bewegung entlang der Z^k-Richtung für den Fall eines nicht zentrierten Objektes dazu, dass der Mittelpunkt des Objektes sich dem Bildmittelpunkt nähert. Diese Bewegung wird mithilfe von $\mathbf{J}_\mathrm{xvz}$ ausgeglichen. Dadurch wird verhindert, dass sich das Objekt aus dem Blickfeld der Kamera bewegt [173]. Dieses Vorgehen impliziert ferner unterschiedliche Schrittweiten $\boldsymbol{\lambda}_\mathrm{e} = [\lambda_\mathrm{vxy}, \lambda_\mathrm{vz}, \lambda_{\omega\mathrm{xy}}, \lambda_{\omega\mathrm{z}}]^\mathrm{T}$. Das vollständige Regelsystem mit Kantenmerkmalen ist in Abb. 5.6 gegeben.

Da bei der Verwendung der langen Kanten als Merkmal die Jacobi-Matrix $\mathbf{J}_{\mathrm{x}\omega\mathrm{x}}$ im Optimum aufgrund der Parallelität der Kanten singulär ist, kann der Gradient bzgl. der Neigung $[\omega_\mathrm{x}, \omega_\mathrm{y}]^\mathrm{T}$ nicht über die Pseudoinverse berechnet werden (vgl. [16]). Daher wird die Orientierung des Ankers verwendet, damit sich die Kamera ausschließlich um Achsen, die orthogonal zur Symmetrieachse des Ankers sind, drehen kann. Es ergibt sich für die Winkelgeschwindigkeit

$$\begin{bmatrix} \omega_\mathrm{x} \\ \omega_\mathrm{y} \end{bmatrix} = -\lambda_{\omega\mathrm{xy}} \cdot \frac{k\mathbf{w}}{||\mathbf{w}||_2}, \text{ mit } \mathbf{w} = \begin{bmatrix} 1 \\ \tan(\pi + \phi_\mathrm{A}) \end{bmatrix}, \tag{5.11}$$

wobei k das korrekte Vorzeichen anzeigt, welches aus ψ_A und $\mathbf{J}_{\psi\omega\mathrm{x}}$ ableitbar ist. Bei der Verwendung der Bodenebene dagegen ist $[\omega_\mathrm{x}, \omega_\mathrm{y}]^\mathrm{T} = -\lambda_{\omega\mathrm{xy}}\mathbf{J}_{\psi\omega\mathrm{x}}^{-1}$ anwendbar.

Um große Schritte bei der Regelung zu vermeiden, werden Bewegungen von mehr als $5\,\mathrm{cm}$ bzw. $0{,}05\,\mathrm{rad}$ abgeschnitten.

5.2.4 Adaptiver digitaler Zoom

Während der bildbasierten Regelung variiert die Größe des Objektes stark. Objekte mit ungünstiger Größe können dann möglicherweise aufgrund der Augmentierung (s. Abschnitt 5.1.3) oder der Unschärfe nach dem Heruntertasten nicht mehr durch das U-Net erkannt werden. Beispielsweise kann zu Beginn der Regelung der Abstand zum Objekt $Z_\mathrm{obj}^\mathrm{K}$ relativ groß sein, um einen möglichst großen Bereich nach Ankern abzusuchen. Dies kann dazu führen, dass das Objekt im Bild (vgl. Gl. (2.8)) zu klein für die Detektion mittels U-Net ist.

Daher ist eine Anpassung der Größe des Objektes erforderlich, die entweder durch die Änderung des Abstandes $Z_\mathrm{obj}^\mathrm{K}$ oder durch digitalen Zoom erreicht wird. Da Ersteres eine Bewegung der Kamera impliziert, wird im Folgenden der digitale Zoom verwendet, um den Bereich, in dem das System skaleninvariant ist, zu vergrößern. Hierfür wird angenommen, dass die Auflösung des Kamerabildes größer ist als die Eingabe in das U-Net.

Das Verfahren wird dann angewendet, wenn im Eingabebild kein Ankerobjekt identifiziert wird (s. Abschnitt 5.1.5). Bei zu kleinen Objekten im Bild wird jedoch die Ankermaske oft hinreichend genau geschätzt, während einzelne Komponenten nicht erkannt werden (s. Abschnitt 6.1.3). Bei zu großen Objekten wird die Ankermaske nur teilweise erkannt, da das rezeptive Feld des neuronalen Netzes zu klein ist [109]. Dies ermöglicht es, die ungefähre Position $[\hat{u}_\mathrm{A}, \hat{v}_\mathrm{A}]^\mathrm{T}$ und die Größe $\hat{s}_\mathrm{A}$ des Ankers im Bild abzuschätzen.

Bei dem hier vorgestellten digitalen Zoom wird iterativ ein Zoomfaktor r_z bestimmt. Ist dieser kleiner als 1, wird statt des gesamten Kamerabildes nur ein Ausschnitt des Bildes um den Punkt $[\hat{u}_\mathrm{A}, \hat{v}_\mathrm{A}]^\mathrm{T}$ heruntergetastet und als Eingabe für das U-Net verwendet. Dies führt dazu, dass das Objekt relativ zum Gesamtbild größer wird. Gilt dagegen $r_\mathrm{z} > 1$, werden

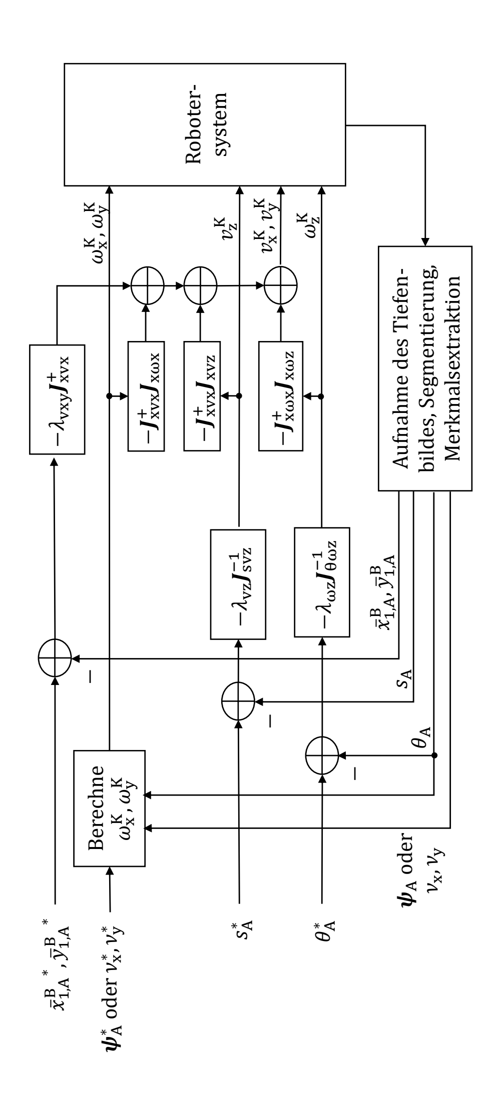


Abbildung 5.6 Regelsystem.

Nullen an das Kamerabild angehängt, bis die Höhe und Breite des Bildes um den Faktor r_z größer ist. Das anschließende Heruntertasten führt dazu, dass der potentielle Anker im Bild kleiner wird.

Beginnend mit dem Zoomfaktor $r_\mathrm{z} = 1$ wird dieser via gleitendem Mittelwert mit Momentum α_z folgendermaßen aktualisiert

$$r_\mathrm{z} \leftarrow \alpha_\mathrm{z} r_\mathrm{z} + (1 - \alpha_\mathrm{z}) \cdot \sqrt{\frac{0{,}1}{\hat{s}_\mathrm{A}}}. \tag{5.12}$$

Dies führt dazu, dass die Objektgröße im Bild gegen den Wert $0{,}1$ konvergiert. Nach jeder Aktualisierung von r_z wird das von der Kamera aufgenommene Bild um Nullen erweitert bzw. zugeschnitten, um die Seitenlängen um den Faktor r_z zu skalieren. Anschließend wird das Bild auf die Eingabegröße des U-Net skaliert, segmentiert und auf vorhandene Anker überprüft. Falls kein Anker gefunden wird, wird r_z erneut aktualisiert.

6 Evaluation der CNNs für die robotergestützte Inspektion

In diesem Kapitel werden das U-Net und der bildbasierte Regler aus Kapitel 5 untersucht. Im ersten Teil (s. Abschnitt 6.1) werden Augmentierungsverfahren und Erweiterungen für das U-Net untersucht und Implikationen hinsichtlich der bildbasierten Regelung diskutiert. Im zweiten Teil (Abschnitt 6.2) wird der bildbasierte Regler evaluiert. Hierfür werden simulativ Voruntersuchungen hinsichtlich der modellbasierten Objekterkennung aus Abschnitt 5.1.5 und des digitalen Zooms aus Abschnitt 5.2.4 durchgeführt. Anschließend wird der Regler bei unterschiedlichen Störeinflüssen untersucht. Am Ende des zweiten Teils wird das vorgestellte Reglerkonzept in einer realen Anwendung evaluiert.

6.1 Ergebnisse der Ankersegmentierung

In diesem Abschnitt werden die Augmentierungsverfahren und Erweiterungen des U-Net aus Abschnitt 5.1 evaluiert. Hierfür wird in Abschnitt 6.1.1 die Simulationsumgebung erläutert. Die Verfahren werden in verschiedenen Kombinationen danach in Abschnitt 6.1.2 verglichen. Anschließend werden die Eigenschaften der gefundenen Methode und deren Einfluss auf die bildbasierte Regelung diskutiert (s. Abschnitt 6.1.3).

6.1.1 Evaluationsumgebung

Für die Bewertung von Augmentierung und Erweiterungen wird jeweils ein CNN von Grund auf mit dem Datensatz aus Abschnitt 5.1.2 trainiert. Jedes Einzelexperiment wird hierbei dreimal wiederholt und das Ergebnis wird gemittelt. Die Größe der Eingabebilder in das U-Net beträgt 224×224 Pixel. Bei jedem Training wird das jeweilige U-Net für 100

Durchläufe mit jeweils 128 Iterationen mit dem NADAM-Optimierer trainiert. Für die Chargengröße gilt $N_{\mathcal{B}} = 16$ aufgrund der Größe der Bilder und des Netzes. Ferner wird ein kosinusartiger Abfall der Lernrate mit einer initialen Lernrate von 10^{-3} verwendet. Neben dem generalisierten Sörensen-Dice-Koeffizienten wird der Jaccard-Koeffizient der einzelnen Komponentenklassen als Vergleichsmetrik verwendet. Als Aktivierung der letzten Schicht wird standardmäßig die Sigmoid-Funktion verwendet. Das Verfahren ist in `python` mithilfe der `tensorflow 2.0`-Bibliothek implementiert.

Um den Einfluss der Augmentierung zu untersuchen, wird zwischen vier verschiedenen Augmentierungsgraden jeweils mit und ohne Hintergrundaugmentierung unterschieden. Mit Hintergrundaugmentierung wird mit der Wahrscheinlichkeit $0{,}5$ ein zufälliger Hintergrund verwendet, ansonsten bleibt das Eingabebild unverändert. Die verschiedenen Grade verwenden die Stufen der Augmentierungskaskade aus Abschnitt 5.1.3. Bei Grad 0, der untersten Stufe, wird keine Augmentierung verwendet. Beim ersten Grad wird eine Basisaugmentierung aus Spiegelung, affinen Transformationen und *Cutout* verwendet. Für Grad II wird Grad I um die letzte Stufe der Augmentierungskaskade (Gaußfilterung, Helligkeitsänderung, ...) erweitert. Für den letzten Grad (Grad III) wird die *Cutout*-Stufe um zwei eigene Verfahren erweitert, bei denen einzelne Komponenten des Ankers verdunkelt oder mehrere kleinere Rechtecke statt eines großen im Bild verdeckt werden. Bei allen *Cutout*-Verfahren wird die Grundwahrheit nicht verändert. Diese wird dazu benutzt, eine geeignete Position des Rechtecks auszuwählen, sodass immer ein Teil des Ankers in diesen Rechtecken liegt. Damit sollen mögliche Verdeckungen simuliert werden.

Mit dem sich so ergebenden besten Augmentierungsgrad werden die Erweiterungen für das U-Net beginnend mit den Aktivierungen untersucht. Anschließend wird der Einfluss von Kanteninformationen analysiert.

6.1.2 Vergleich der Verfahren

Die Testergebnisse für die unterschiedlichen Grade der Augmentierung mit und ohne Hintergrundaugmentierung (Hgrd.) sind in Tab. 6.1 abgebildet. Diese zeigt den mittleren Jaccard-Koeffizienten der einzelnen

Ankerkomponenten sowie der Ankermaske und den mittleren gSDK sowie dessen Standardabweichung (std). Durch das Weglassen teilverdeckter Anker beim Trainings- und Testdatensatz verbessern sich die Ergebnisse im Vergleich zu [186].

Tabelle 6.1 Die Ergebnisse für die unterschiedlichen Grade der Augmentierung.

Hgrd.	nein				ja			
Grad	0	I	II	III	0	I	II	III
Anker	0,724	0,864	0,869	0,869	0,869	**0,950**	0,945	0,949
Kom.	0,370	0,679	0,710	0,704	0,684	0,733	**0,742**	0,740
Welle	0,237	0,529	0,554	0,548	0,543	0,666	0,656	**0,673**
Ritzel	0,203	0,339	0,365	0,363	0,325	0,483	0,503	**0,507**
gSDK	0,364	0,193	0,185	0,184	0,187	0,096	0,096	**0,092**
std	0,015	0,006	0,006	0,001	0,003	0,002	0,002	0,003

Die deutlichsten Verbesserungen können durch die Verwendung der Hintergrundaugmentierung und durch die affinen Transformationen erzielt werden. Für beide Verfahren wird der gSDK um ca. 50 % verringert. Die Verbesserung durch Stufe II und Stufe III ist unter Bezugnahme zur gemessenen Standardabweichung gering, aber vorhanden. Insgesamt liefert die Augmentierung des Grades III mit Hintergrundaugmentierung die besten Ergebnisse. Daher wird diese Augmentierungsstrategie im Folgenden verwendet.

Die Testergebnisse für die verschiedenen Aktivierungen und Kanteninformationen sind Tab. 6.2 dargestellt. Die *softmax*-basierte Aktivierung (MSM) führt in allen Fällen zu einer Verschlechterung des gSDK im Bereich von 50 %, was v. a. auf die schlechte Konvergenzeigenschaft der Ankerkomponente *Welle* zurückzuführen ist. Dies ist in der deutlich höheren Standardabweichung des gSDK ebenfalls ablesbar. Gegenüber der einfachen *sigmoid*-Aktivierung (Sig) verschlechtert sich der gSDK der multiplikativen *sigmoid*-Aktivierung (MSig) um ca. 2,3 %. Trotz der Beseitigung von logischen Widersprüchen führen die Aktivierungen, die diese herbeiführen, zu einer Verschlechterung des gSDK. Das U-Net ist somit in der Lage, die logischen Zusammenhänge während des Trainings selbständig zu erlernen.

Tabelle 6.2 Die Ergebnisse für die verschiedenen Aktivierungen des U-Net und bei der Verwendung von Kanteninformationen via Addition (A) und Konkatenation (K).

Kante	-			A			K		
Aktiv.	Sig	MSig	MSM	Sig	MSig	MSM	Sig	MSig	MSM
Anker	0,949	0,947	0,939	0,948	0,947	0,944	**0,951**	0,948	0,944
Kom.	0,740	0,740	0,733	0,748	**0,755**	0,738	0,750	0,748	0,740
Welle	0,673	0,667	0,092	0,663	0,669	0,093	**0,687**	0,676	0,094
Ritzel	**0,507**	0,500	0,467	0,496	0,494	0,468	0,498	0,504	0,470
gSDK	0,092	0,096	0,145	0,095	0,094	0,141	**0,090**	0,093	0,138
std	0,002	0,001	0,008	0,010	0,005	0,007	0,004	0,002	0,006

Zusätzliche Kanteninformationen können die Segmentierung verbessern, jedoch führt eine Addition zu einer deutlich höheren Standardabweichung des Ergebnisses. Bei einer Konkatenierung dagegen ist die Standardabweichung geringer und die Verbesserung des gSDK ist mit ca. 2,7 % gegenüber 0,7 % bei Addition eindeutiger. Diese Beobachtungen kulminieren darin, dass eine *sigmoid*-Aktivierung mit einer Kantenkonkatenierung die besten Ergebnisse liefert.

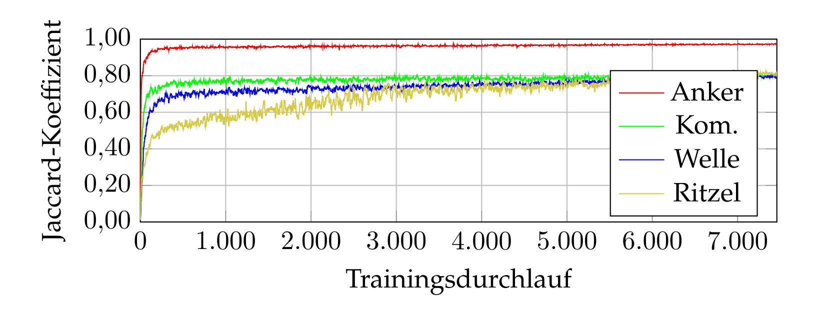


Abbildung 6.1 Jaccard-Koeffizient der Validierungsdaten während des Trainings.

6.1.3 Eigenschaften der Segmentierung

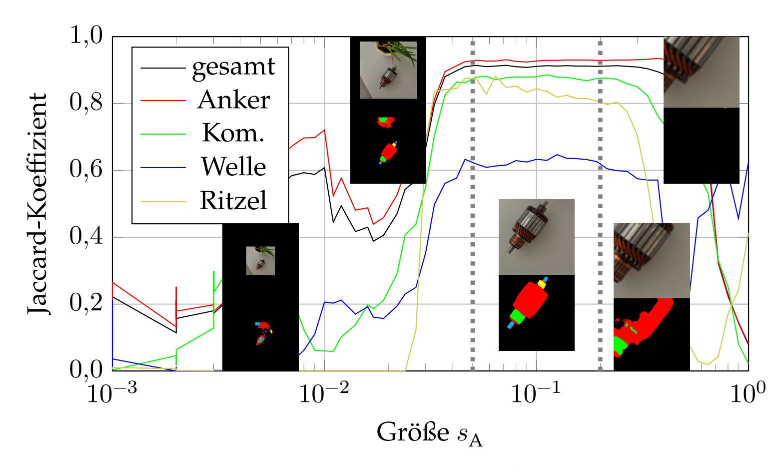
Mit den Ergebnissen aus Abschnitt 6.1.2 wird nun das U-Net 7.500 Durchläufe trainiert. Der resultierende gSDK beträgt 0,0656 und die Jaccard-Koeffizienten sind 0,972 (A), 0,796 (K), 0,789 (W) und 0,811 (R). Auffal-

lend beim Training ist, dass der Jaccard-Koeffizient der Ankermaske $\mathbf{P}_\mathrm{A}$ und der Kommutatormaske $\mathbf{P}_\mathrm{K}$ sehr schnell in Sättigung gehen, während sich der Jaccard-Koeffizient von Ritzel und Welle näherungsweise linear mit der Anzahl der Durchläufe ab etwa dem 500. Durchlauf vergrößert (s. Abb. 6.1).

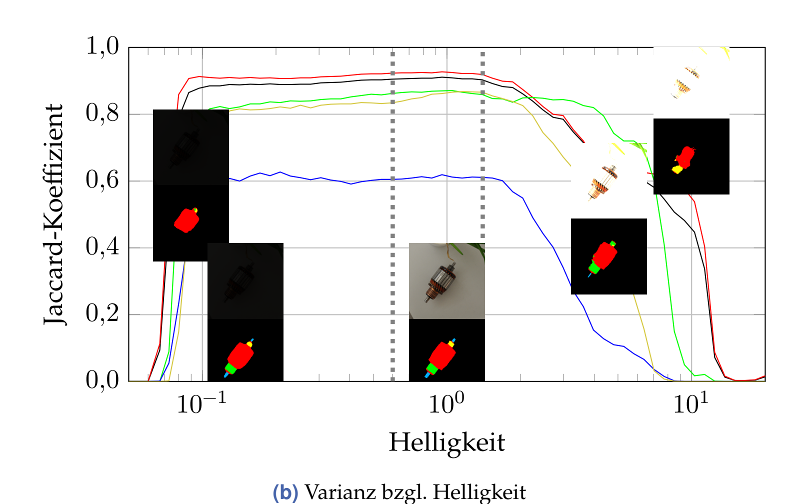
Im Folgenden werden die Invarianzeigenschaften der Segmentierung untersucht. Hierzu werden der gSDK und die Jaccard-Koeffizienten für unterschiedliche Größen und Orientierungen des Ankers sowie für unterschiedlichen Kontrast und Helligkeit des Eingabebildes untersucht. Dabei werden für jeden Wert insg. 256 Bilder aus dem Eingabebild erzeugt, die sich lediglich hinsichtlich einer Translation unterscheiden. Für Kontrast, Helligkeit und Orientierung wird das Bild derart skaliert, dass die normierte Ankergröße $s_\mathrm{A} = 0{,}06$ ist. Die Werte für gSDK und Jaccard-Koeffizienten weichen im Vergleich zu den oben genannten Ergebnissen ab, da nur ein einziges Bild untersucht wird.

Das Ergebnis für die Ankergröße und die Helligkeit ist in Abb. 6.2 abgebildet. Die senkrechten gepunkteten grauen Linien entsprechen den Grenzen der jeweiligen Operation bei der Augmentierung. In beiden Fällen ist das Segmentierungsergebnis innerhalb dieser Grenzen näherungsweise konstant. Dies entspricht den Erwartungen.

Übersteigt die Ankergröße den Wert von $0{,}2$ steigt der gSDK, da die Größe das rezeptive Feld des U-Net übersteigt, infolgedessen wird der Anker nur in Teilen bzw. gar nicht erkannt. Bei sinkender Ankergröße fällt der Jaccard-Koeffizient von Kommutator, Welle und Ritzel steil ab, sobald die untere Grenze von $0{,}04$ unterschritten wird. Die Ankermaske kann weitestgehend stabil detektiert werden. Jedoch führt ein zweites Objekt im Bild zu einer Fehldetektion, die den Jaccard-Koeffizienten negativ beeinflusst, die aber durch ein Modell wie das aus Abschnitt 5.1.5 entfernt werden kann. Insgesamt kann die Ankermaske $\mathbf{P}_\mathrm{A}$ auch bei Werten unter $0{,}005$ detektiert werden, was eine Positionsbestimmung und die Schätzung der ungefähren Größe des Ankers ermöglicht.



(a) Varianz bzgl. Ankergröße

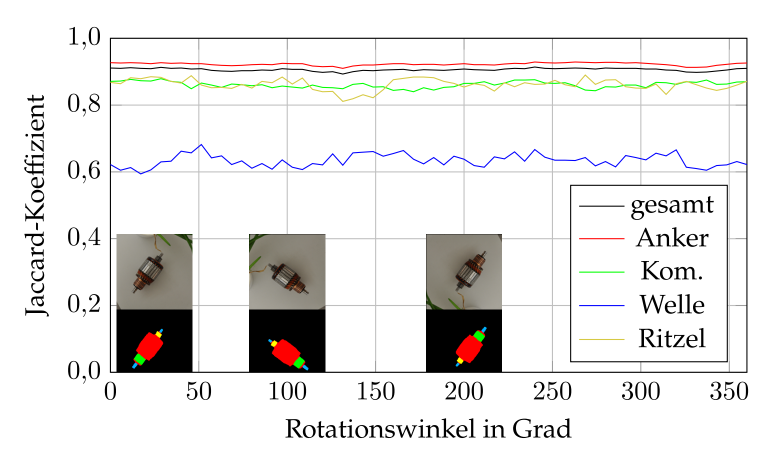


(b) Varianz bzgl. Helligkeit

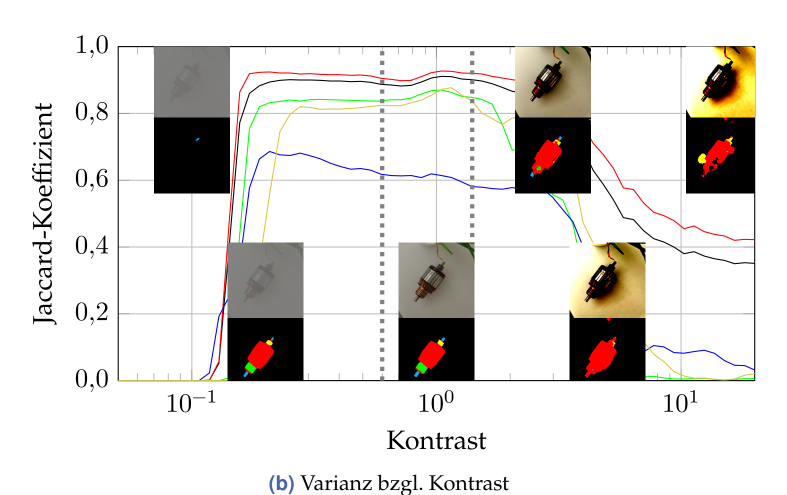
Abbildung 6.2 Segmentierungsvarianz bzgl. Größe des Ankers und Helligkeit. Mit *gesamt* ist der mittlere Jaccard-Koeffizient gemeint, der mit $1 - \text{gSDK}$ korrespondiert.

Trotz des geringen Augmentierungsbereiches bei der Helligkeit kann der Anker über einen Bereich von 0,03 bis 10 gut erkannt werden. Wie die in den Graphen eingefügten Bilder zeigen, kann der Anker auch

bei extrem dunklen und stark überbelichteten Bildern erkannt werden. Aus Abb. 6.3 kann ein ähnliches Verhalten für den Kontrast abgeleitet werden. Ebenso zeigt Abb. 6.3, dass das trainierte U-Net weitestgehend rotationsinvariant ist.



(a) Varianz bzgl. Orientierung.



(b) Varianz bzgl. Kontrast

Abbildung 6.3 Segmentierungsvarianz bzgl. Orientierung des Ankers und Kontrast.

Insgesamt zeigt das U-Net Eigenschaften bzgl. der Invarianz, die die Erwartungen gemäß der Augmentierung übersteigen. Für die Bildmanipulationen, die mit Augmentierungsgrad II hinzu kommen (s. Abschnitt 6.1.2), ist die Segmentierung bis in Regionen, bei denen es für das menschliche Auge schwierig wird, weitestgehend stabil (s. Abb. 6.2). Bei den affinen Transformationen behält die Segmentierung innerhalb der vordefinierten Grenzen seine Invarianz. Bei einer Verschlechterung der Segmentierung ist die Schätzung der Ankermaske $\mathbf{P}_\mathrm{A}$ im Gegensatz zu den anderen Komponenten deutlich robuster. Dies rechtfertigt die Definition der Ankerklassen nach Abschnitt 5.1.2 und kann bei der bildbasierten Regelung, falls der Anker in einer Aufnahme nicht erkannt wird, einen entscheidenden Vorteil mit sich bringen, da eine ungefähre Position und Größe des Ankers ermittelt werden kann, die als Grundlage für den in Abschnitt 5.2.4 vorgestellten digitalen Zoom dient.

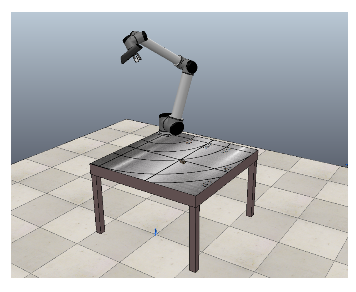
6.2 Evaluation des Reglers

In diesem Abschnitt wird der Regler mit seinen Erweiterungen untersucht. Nachdem in Abschnitt 6.2.1 die Simulationsumgebung beschrieben wird, wird in Abschnitt 6.2.2 das Verhalten des Merkmalsextraktors thematisiert, wenn die Ausgabe des U-Net mit den in Kapitel 5 beschriebenen Erweiterungen für die Segmentierung verwendet wird. Anschließend wird die Robustheit des Reglers simulativ gegenüber der Anfangsposition (s. Abschnitt 6.2.3), gegenüber stochastischen Störungen (s. Abschnitt 6.2.4) und beim Auftreten mehrerer Objekte untersucht (s. Abschnitt 6.2.5). Anschließend wird in Abschnitt 6.2.6 der Regler in einem realen System evaluiert.

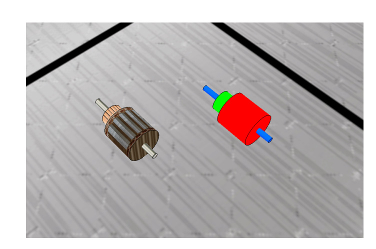
6.2.1 Simulationsumgebung

Das System wird mit `CoppeliaSim` simuliert. Hierzu wird als Roboterarm ein `UR-10` mit einer `Kinect`-Tiefenbildkamera an dessen Endeffektor verwendet. Der Roboter wird über die `python remote API` gesteuert und die Implementierung des U-Net erfolgt gemäß Abschnitt 6.1.1. Die Auflösung der `Kinect` beträgt 960×960 Pixel nach dem Zuschneiden zum Quadrat. Die Parameter der Kameramatrix $\mathbf{C}_\mathrm{K}$

sind aus dem Simulationsaufbau extrahierbar und die Kamera hat eine unendliche Schärfentiefe. Der Tiefensensor liefert für jedes Bildpixel eine fehlerfreie Tiefe. Die Abtastzeit der Simulation beträgt 50 ms.



(a) Gesamtaufbau



(b) Modelle der Anker

Abbildung 6.4 In der Simulation befinden sich der Roboter und der Anker auf einem quadratischen Tisch mit der Seitenlänge 1,1 m. In der rechten Grafik sind beide Modelle der Anker dargestellt. Der rechte Anker wird als Grundwahrheit genutzt.

Die Bilder der Kamera werden für die Eingabe in das U-Net auf 336×336 Pixel heruntergetastet. Dies führt dazu, dass sich die Reichweite der Detektierbarkeit zu kleineren Ankergrößen verschiebt, da das U-Net mit Bildern der Auflösung 224×224 trainiert wird. Das U-Net wurde gemäß Abschnitt 6.1.3 trainiert. In der Simulation wird eine Zusammensetzung mehrerer Zylinder mit geeigneter Textur als Anker verwendet. Als Grundwahrheit werden die Zylinder gemäß der Konvention aus Abb. 5.2 eingefärbt. Durch geeignete Farbfilter ist die Grundwahrheit direkt aus dem aufgenommenen Bild ableitbar. Beide Anker haben dieselben Maße. Der Gesamtaufbau und die beiden verwendeten Ankermodelle sind in Abb. 6.4 illustriert.

6.2.2 Ankermodell und digitaler Zoom

In diesem Abschnitt wird die modellbasierte Segmentierung (s. Abschnitt 5.1.5) und der digitale Zoom (s. Abschnitt 5.2.4) simulativ untersucht. Die Koordinatenangaben werden in Metern angeben. Hierfür wird mit der Kamera eine Trajektorie um den Anker, der sich am Punkt $\mathbf{X}_{\text{obj}}^{(W)} = [0{,}5; 0{,}5; 0]^{\mathrm{T}}$ auf der Tischplatte befindet, gefahren, die aus den folgenden insg. 240 Schritten besteht:

- Beginnend bei $\mathbf{X}^{(W)} = [0{,}5; 0{,}5, 0{,}35]^T$ entfernt sich die Kamera vom Objekt bis zum Punkt $\mathbf{X}^{(W)} = [0{,}5; 0{,}5; 1{,}05]^T$. Anschließend nähert sich die Kamera bis auf $\mathbf{X}^{(W)} = [0{,}5; 0{,}5; 0{,}1]^T$ dem Anker, um anschließend zu seiner Ausgangsposition zurückzufahren (Schritt 0 bis 48). Hierdurch wird die Größe des Ankers variiert, was sich insbesondere für das Testen des digitalen Zooms eignet.

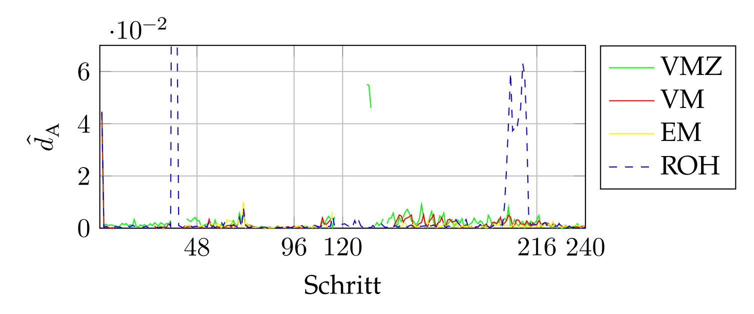
- Die Kamera dreht sich 360° um die z-Achse, um die Orientierung des Ankers im Bild zu beeinflussen (Schritt 48 bis 96).

- Die Kamera wird auf den Punkt $\mathbf{X}^{(W)} = [0{,}0; 0{,}0; 0{,}15]^T$ bewegt (Schritt 96 bis 120) und fährt anschließend einen 3/4-Kreis, der parallel zur Tischplattenebene ist, um den Anker (Schritt 120 bis 216) und danach einen 1/4-Kreis senkrecht zur Bodenebene, um wieder zur Ausgangsposition der Trajektorie zurückzukehren (Schritt 216 bis 240). Hierbei werden die problematischen Positionen angefahren, für die das Modell aufgrund eines verdeckten Kommutators keinen Anker erkennen kann.

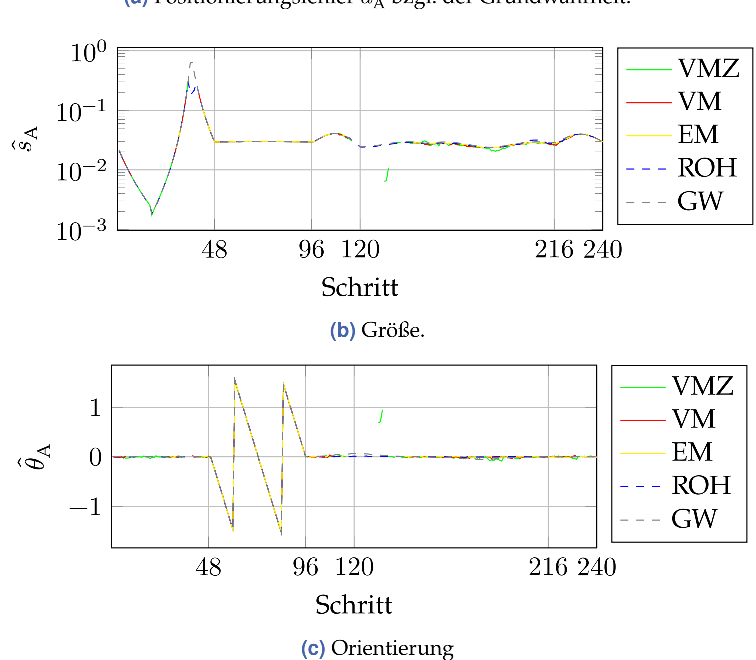
Für die Versuche werden die geschätzten Merkmale Position, Größe, Neigung bzgl. Bodenplatte, Neigung als Kantenmerkmal und Orientierung mit der beschriebenen Grundwahrheit (GW) verglichen. Die Merkmale werden mit der Rohausgabe des U-Net (ROH), einem einfachen Modell für die Erkennung des Ankers anhand der Komponenten (EM) und dem vollständigen Modell gemäß Abschnitt 5.1.5 (VM) verglichen. Letzteres wird zusätzlich mit dem digitalen Zoom aus Abschnitt 5.2.4 untersucht (VMZ). Die geschätzten Merkmale sind in Abb. 6.5 dargestellt.

Bei der Verwendung der Modelle treten bei der Schätzung der Merkmalsparameter Ausfälle auf, die auf das Nicht-Erkennen eines Objektes als Anker zurückzuführen sind. Ferner werden Schätzungen mit starker Abweichungen der Position und der Größe als Ausfall gewertet. Hieraus ergibt sich für ROH eine Ausfallrate von 4,17 %, bei EM 28,75 %, bei VM 22,50 % und bei VMZ 11,25 %. Zwar ist die Ausfallrate bei der Rohausgabe am geringsten, jedoch führen Fehldetektionen zu starken Ausreißern bei den Merkmalen, die nicht durch das Modell unterdrückt werden und somit direkt in die Regelung einfließen. Weitere negative Eigenschaften der Rohausgabe treten beim Vorhandensein mehrerer Objekte auf (s. Abschnitt 6.2.5). Bei der Verwendung eines Modells treten keine Ausfälle

auf, die auf fehlerhafte Größen- oder Positionsschätzung zurückzuführen sind. Ferner werden keine Merkmale geschätzt, falls kein Anker im Bild gefunden wird. Je detaillierter das Modell, desto geringer ist die Ausfallrate.

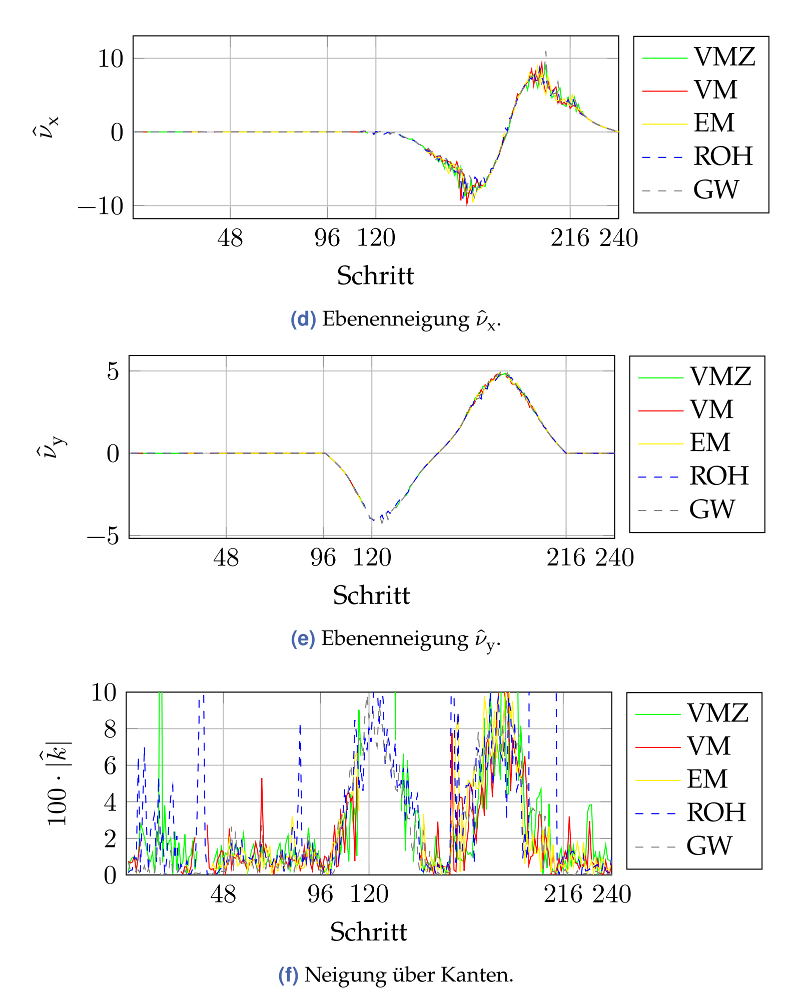


(a) Positionierungsfehler d_A bzgl. der Grundwahrheit.



(b) Größe.

(c) Orientierung



(d) Ebenenneigung $\hat{\nu}_x$.

(e) Ebenenneigung $\hat{\nu}_y$.

(f) Neigung über Kanten.

Abbildung 6.5 Geschätzte Parameter für die Grundwahrheit (GW), die Rohausgabe (ROH), das einfache Modell (EM) und das vollständige Modell mit (VMZ) bzw. ohne Zoom (VM).

Die Ausfälle bei den Modellen treten u. a. dann auf, wenn die optische Achse parallel zur Symmetrieachse des Ankers steht und der Kommutator verdeckt ist. Dies ist zwischen Schritt 110 und 150 der Fall. Ein fälschlich detektierter Anker bei VZM führt in diesem Bereich zu ei-

nem Ausreißer der geschätzten Merkmale. Weitere Ausfälle der Modelle sind bei ungünstigen Ankergrößen zu beobachten (s. Abb. 6.5(b)). Bei zu kleinen Objektgrößen, d. h. $\hat{s}_{\mathrm{A}} < 10^{-2}$, wird ein Objekt nur durch ROH und VMZ erkannt. Da VM und EM in diesem Bereich nichts erkennen, werden in der Rohausgabe des U-Net nicht alle notwendigen Ankerkomponenten erkannt, weswegen formal kein Anker im Bild sichtbar ist. Durch den digitalen Zoom kann der Anker derart vergrößert werden, dass der Anker mit allen Komponenten erkannt wird. Bei zu großen Objekten kann der digitale Zoom bis zu einem gewissen Grad die Detektion des Ankers aufrechterhalten, aber er versagt, sobald ROH das Objekt nur noch teilweise erkennt. Diese Beobachtung deckt sich mit den Ergebnissen aus Abschnitt 6.1.3. In der Praxis werden diese Größen nur beim Unterschreiten des technisch bedingten Mindestabstandes erreicht.

Insgesamt hat VM eine höhere Toleranz gegenüber der Perspektive als EM, was zu weniger Ausfällen führt. Hintergrund ist, dass durch die Änderung des Bildes nach einem Schritt in manchen Fällen bestimmte Komponenten nicht erkannt werden. Dies ist aber durch das Modell nach Abschnitt 5.1.5 korrigierbar.

Außerhalb dieser beiden kritischen Bereiche unterscheiden sich die Schätzungen der Ankerparameter bis auf die oben beschriebenen Ausreißer von ROH kaum, da die erkannten Anker ähnlich sind.

Die geschätzten Ebenenneigungen des Tisches sind für alle untersuchten Fälle nahezu identisch, da die Schätzung weitestgehend unabhängig von der Erkennung des Ankers ist. Bei höheren Neigungen ist die Schätzung verrauschter, da der Anker einen größeren Teil des Tisches verdeckt.

Da der Anker parallel zur X^{W}-Achse orientiert und rotationssymmetrisch ist, ist das Kantenmerkmal unabhängig von der Tischneigung ν_x und korreliert bis auf einige Ausreißer mit dem Betrag der Neigung ν_y. Zwar wird das Kantenmerkmal der Rotationssymmetrie des Ankers gerecht, aber das Merkmal ist deutlich verrauschter und abhängig vom detektierten Anker, während die Neigung der Tischplatte unabhängig davon ist. Eingedenk der höheren Ausfallraten bei ungünstiger Perspektive, die insbesondere bei starker Neigung gegeben ist, ist die Regelung über die Neigung der Tischebene vielversprechender.

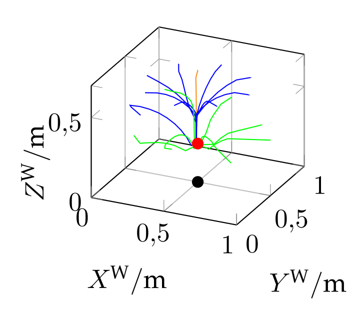
6.2.3 Einfluss gegenüber der Initialposition

Im Folgenden wird der Regler untersucht. Zunächst wird in diesem Abschnitt der Einfluss der Initialposition des Reglers analysiert. Hierbei wird das vollständige Modell mit digitalen Zoom (VMZ) für die Segmentierung verwendet. Die Regelung mithilfe der Kantenmerkmale k_{A} wird mit der Regelung mittels Neigungen $[\nu_x, \nu_y]^{\mathrm{T}}$ der Tischebene verglichen.

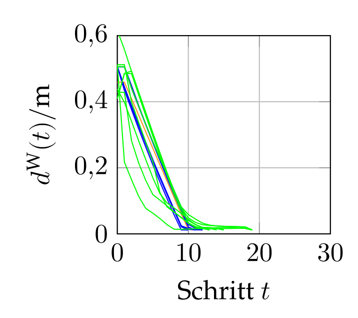
Die verschiedenen Startpositionen befinden sich bei 30°, 60° und 90° Elevation bzgl. des Ankers und bei 45° Schritten des Azimuthwinkels in einer Entfernung von $0{,}5\,\mathrm{m}$ zum Anker, falls diese Position für den Roboterarm möglich ist. Die Ankerposition ist wie in Abschnitt 6.2.2. Die Regelung wird abgebrochen, falls ein gewichtetes Fehlermaß einen Schwellwert unterschreitet. Die Ergebnisse sind in Abb. 6.6 dargestellt. Die leicht unterschiedlichen Anfangspositionen in den Diagrammen resultieren aus dem Einschwingvorgang und dem ersten Schritt des Reglers.

Der Hauptunterschied beider Regelungsverfahren besteht darin, dass sich die möglichen Zielpositionen beim Kantenmerkmal auf einem Kreis befinden, während beim Ebenenmerkmal die Zielposition ein einziger Punkt ist. Das Ebenenmerkmal ist damit nur anwendbar, falls die Symmetrieachse des Ankers parallel zur Tischebene ist.

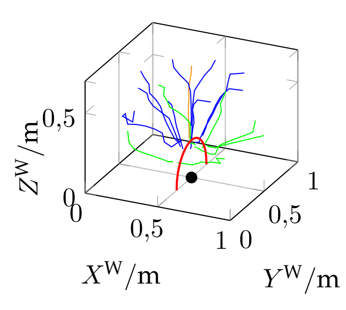
Gemäß Abb. 6.6 sind die Trajektorien des Ebenenmerkmals glatt und der Positionierungsfehler d^{W} konvergiert in 10 Schritten für 60° und 90° Elevation und in max. 20 Schritten bei 30°. Dagegen haben die Trajektorien des Kantenmerkmals einen zackenförmigen Verlauf. Dieser ist auf die verrauschte Schätzung des Kantenmkermals (s. Abb. 6.5) zurückzuführen. Bei der Mehrheit der Startpositionen konvergiert der Regler jedoch in einer ähnlichen Geschwindigkeit und bei 30° Elevation teilweise sogar schneller. Die Ausreißer treten in den Fällen auf, in denen zu Beginn der Kommutator des Ankers nicht sichtbar ist oder es aufgrund des flachen Elevationswinkels zu einer Kollision mit der Tischplatte kommt. Letzteres hat zu einem Ausfall geführt, bei dem der Regler sich nicht wieder in eine geeignete Position manövrieren konnte. Daher sind nicht alle Positionen auf dem Kreis sinnvolle Zielpunkte.



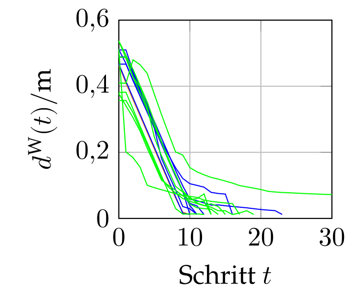
(a) Trajektorie mit Ebenenmerkmal.



(b) Positionsfehler d^W beim Ebenenmerkmal



(c) Trajektorie mit Kantenmerkmal.



(d) Positionsfehler d^W beim Kantenmerkmal.

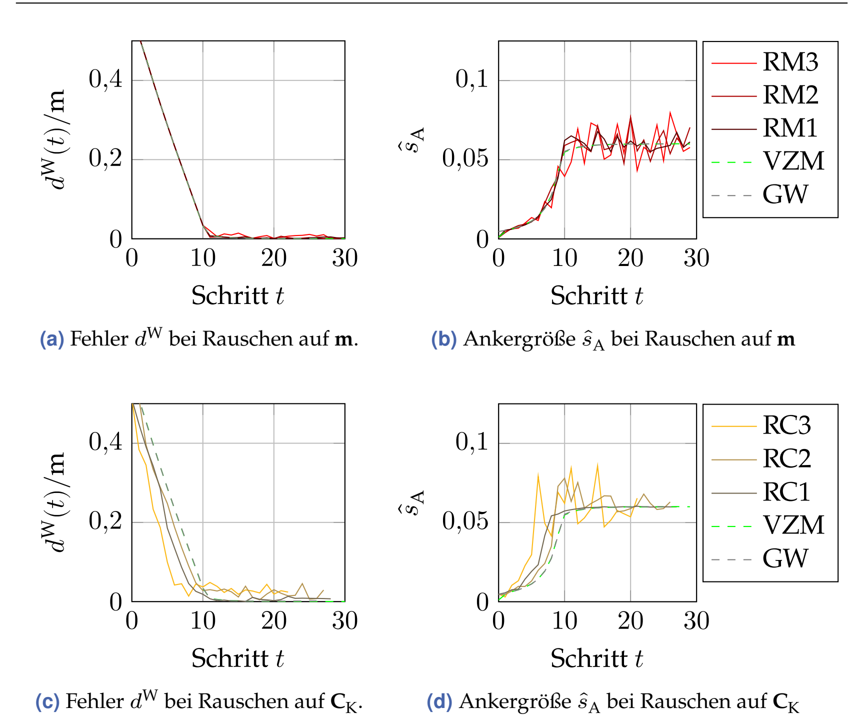
Abbildung 6.6 Trajektorien und Distanz d^W zur gesuchten Position (rot) bei verschiedenen Startpositionen. Farben gemäß dem Elevationswinkel der Startposition (grün: 30°, blau: 60°, gelb: 90°) bzgl. des Ankers (schwarzer Punkt).

Da die Regelung mit dem Ebenenmerkmal dazu führt, dass die optische Achse des Kamerasystems im Optimum orthogonal zur Tischplatte steht, wird eine Kollision mit dem Tisch automatisch vermieden. Außerdem kann der Regler zu Beginn, ohne einen Anker zu detektieren, die Kamera zu einem höheren Elevationswinkel regeln. Somit werden ungünstige Perspektiven auf den Anker implizit vermieden. Ferner kann das Kantenmerkmal für ungünstige Bauformen des Ankers scheitern. Dies passiert insbesondere dann, wenn der Kommutator mit dem dickeren Korpus des Ankers aus der Perspektive der Kamera eine Flucht bildet.

Insgesamt zeigt die Nutzung der Ebenenmerkmale deutliche Vorteile gegenüber den Kantenmerkmalen. Daher wird im Folgenden mithilfe der geschätzten Bodenebene geregelt. Das Kantenmerkmal hat allerdings ein berechtigtes Nischendasein, falls die Annahme der parallelen Bodenebene nicht gegeben ist.

6.2.4 Einfluss von Rauschen

In diesem Abschnitt wird die Robustheit des Reglers gegenüber Rauschen betrachtet, welches künstlich auf die Kameraparameter, die Kameraposition und die geschätzten Merkmale gegeben wird. Verglichen wird die Grundwahrheit (GW) mit dem vollständigen Modell mit digitalen Zoom (VMZ) bei unterschiedlichen Rauschpegeln. Bei divergenten Reglern wird die Regelung nach 30 Schritten abgebrochen. Für die Kameraparameter und die geschätzten Merkmale wird eine gleichverteilte Zufallsvariable auf die geschätzten Werte multipliziert. Dabei werden drei unterschiedlich starke Stufen verwendet, sodass der maximale Fehler 10 % (RC1, RM1), 20 % (RC2, RM2) oder 30 % (RC3, RM3) des wahren Wertes beträgt. Für die Kameraposition wird ein additiver mittelwertfreier normalverteilter Fehler ebenfalls in drei verschiedenen Ausprägungen simuliert. Die Standardabweichungen betragen 0,5 cm und $0{,}3^\circ$ (RP1), 1,5 cm und $0{,}9^\circ$ (RP2) sowie 5,0 cm und $2{,}8^\circ$ (RP3). Die Startposition der Kamera ist im Abstand von 0,5 m zum Anker bei einem Elevationswinkel von 60°.



(a) Fehler d^{W} bei Rauschen auf **m**.

(b) Ankergröße $\hat{s}_{\mathrm{A}}$ bei Rauschen auf **m**

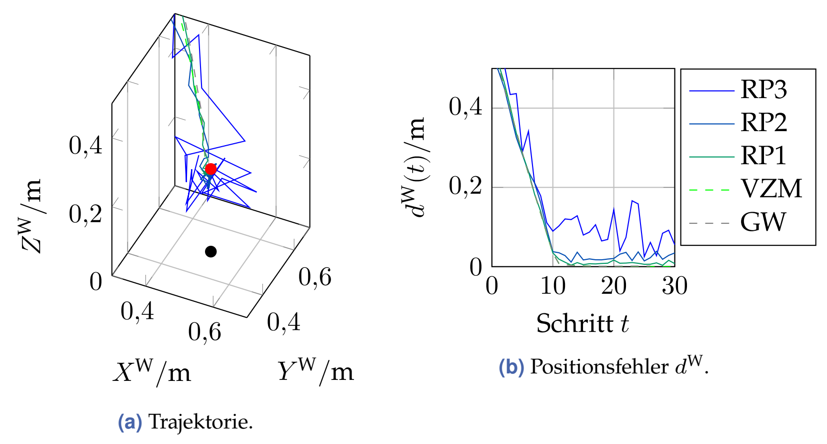
(c) Fehler d^{W} bei Rauschen auf $\mathbf{C}_{\mathrm{K}}$.

(d) Ankergröße $\hat{s}_{\mathrm{A}}$ bei Rauschen auf $\mathbf{C}_{\mathrm{K}}$

Abbildung 6.7 Fehler d^{W} und geschätzte Ankergröße $\hat{s}_{\mathrm{A}}$ bei unterschiedlich starkem Rauschen auf den Merkmalsvektor **m** und die Kameramatrix $\mathbf{C}_{\mathrm{K}}$.

Die Ergebnisse für das Rauschen auf den Merkmalsvektor **m** und die Kameramatrix $\mathbf{C}_{\mathrm{K}}$ sind in Abb. 6.7 dargestellt. Für beide Varianten konvergiert der Regler bei allen Rauschgraden. Das Rauschen führt jeweils zu einem Fehler in der Merkmalsschätzung. Während ein Rauschen auf den Merkmalsvektor diesen direkt und über die Jacobi-Matrix (vgl. Abschnitt 5.2.2) indirekt beeinflusst, wird über ein Rauschen auf $\mathbf{C}_{\mathrm{K}}$ die Transformation von Sensorvariablen in die normierten Bildkoordinaten verfälscht. Je größer der Merkmalsparameter, desto größer ist das Rauschen. Daher ist der Einfluss auf die geschätzte Größe, für die im Optimum $s_{\mathrm{A}}^{*} \neq 0$ gilt, am größten. Der destruktive Einfluss des Rauschens auf den Merkmalsvektor führt hier zu einem kleinen Fehler, der die Konvergenz in der Nähe des Optimums verlangsamt. Dagegen führt das Rauschen auf die Kameraparameter zu einem systematischen Fehler,

wodurch die Endposition der Kamera am Ende wenige Zentimeter vom gesuchten Optimum entfernt ist. Insgesamt treten beide Fehler erst bei enorm großen relativen Fehlern von über 20 % auf. Somit ist der Regler weitestgehend robust gegenüber Fehlern bei der Kamerakalibrierung und gegenüber fehlerhaften Schätzungen des Merkmalsvektors, die bspw. bei Fehlern infolge ungenauer Segmentierung auftreten.



(b) Positionsfehler d^W.

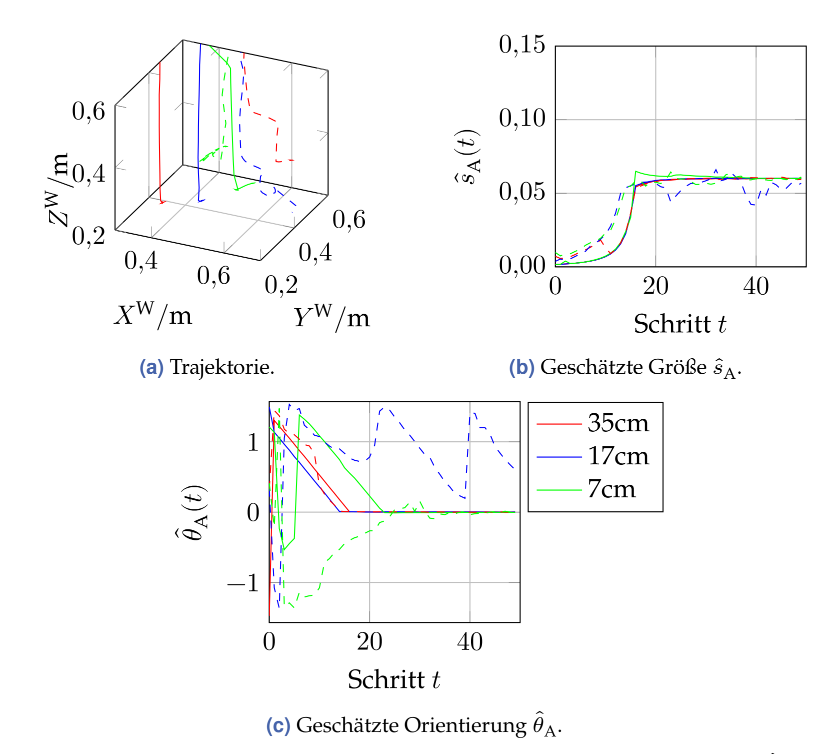
(a) Trajektorie.

Abbildung 6.8 Trajektorie und Positionsfehler d^W bei verrauschter Kameraposition.

Eine verrauschte Kameraposition beeinflusst die Positionsbestimmung der Kamera durch den Regler, da dies gleichbedeutend mit einem Bewegungsvektor $[\mathbf{v}^K(t), \boldsymbol{\omega}^K(t)]^T$ ist, der eine Zufallskomponente enthält. Ist der hierdurch verursachte Fehler bzgl. der Merkmale größer als die Fehlertoleranz des Reglers, kommt es zu nicht konvergierenden Bewegungen um die Zielposition, die nur durch Zufall in den Toleranzbereich des Zielpunktes münden. Dieser Effekt ist in Abb. 6.8 zu beobachten. Die nicht konvergierenden Bewegungen finden in einem Bereich um die Zielposition statt, der sich näherungsweise als Kugel mit dem Radius der dreifachen Standardabweichung des Positionsrauschen bewegt. Dies sind bei R3 15 cm. Insgesamt ist der Regler jedoch robust gegenüber diesem Rauschtyp, sofern der Fehler nicht in derselben Größenordnung wie der Toleranzbereich für die Zielposition ist.

6.2.5 Multiple Objekte

In diesem Abschnitt wird der Fall betrachtet, dass sich mehrere Anker in der Szene befinden. Hierfür wird der Regler mit (VM) und ohne (ROH) Ankermodell verglichen. Die Abstände zum Zielobjekt in diesem Versuch machen keinen digitalen Zoom erforderlich. In der Szene werden drei Anker in einem gleichseitigen Dreieck positioniert. Der Abstand zwischen den Ankern beträgt 35 cm, 17 cm und 7 cm. Die Regelung wird pauschal nach 50 Schritten abgebrochen. Die Ergebnisse sind in Abb. 6.9 dargestellt.



(a) Trajektorie.

(b) Geschätzte Größe $\hat{s}_A$.

(c) Geschätzte Orientierung $\hat{\theta}_A$.

Abbildung 6.9 Trajektorie, geschätzte Größe $\hat{s}_A$ und geschätzte Orientierung $\hat{\theta}_A$ des Ankers für verschiedene Abstände. Bei durchgezogenen Linien wird das Modell nach Abschnitt 5.1.5 verwendet. Die gestrichelten Linien verwenden die Rohausgabe des U-Net.

Unter Verwendung des Modells wird immer der größte detektierte Anker für die Regelung ausgewählt, während die anderen beiden ignoriert werden. Dies kann zu abrupten Richtungsänderungen in der Trajektorie führen, falls ein anderer Anker als größter erkannt wird. Dies geschieht insbesondere beim kleinsten untersuchten Abstand zwischen den Ankern, während bei den größeren beiden ein Anker frühzeitig ausgewählt wird. In allen drei Fällen konvergiert hier der Regler.

Ohne Modell scheint der Regler ebenfalls für die Abstände 7 cm und 35 cm zu konvergieren. Allerdings ist nur für 35 cm eine echte Konvergenz festzustellen, da ein Anker ausgesucht wird und die anderen beiden Anker aus dem Bild verschwinden. Für einen Abstand von 7 cm werden die drei Objekte als ein großer disjunkter Anker interpretiert. Infolgedessen können Merkmale geschätzt werden, deren Fehler durch den Regler minimiert werden. Jedoch führt diese Minimierung nicht zu der gewünschten Positionierung der Kamera. Bei einem Abstand von 17 cm konvergiert der Regler nicht, da die Schätzung der Orientierung keine sinnvollen Ergebnisse liefert, da durch die Konstellation der Objekte die geschätzte Orientierung abhängig von der Blickrichtung der Kamera ist.

Dieser Versuch zeigt, dass das Ankermodell in Fällen, in denen mehrere Anker im Bild sichtbar sind, der Rohausgabe des U-Net deutlich überlegen ist. Dieser Effekt ist noch deutlicher, wenn fremde Objekte in der Szene enthalten sind, für die fälschlicherweise eine Ankermaske geschätzt wird, die aber keine Komponenten enthält, da diese durch das Ankermodell ignoriert werden.

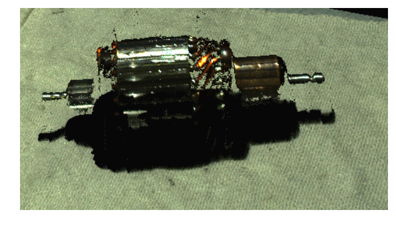
6.2.6 Evaluation des realen Systems

In diesem Abschnitt wird der Einsatz des Reglers in einer realen Umgebung untersucht. Hierzu wird ein `UR10`-Roboter mit einer `Zivid One Plus S` Kamera am Endeffektor verwendet, die Tiefeninformationen über Streifenlichtprojektionen schätzt. Der Roboterarm wird über die `urx python`-Bibliothek nach der IEEE-Norm 802.3 (Ethernet) angesteuert. Die Kamera wird über die `zivid python`-Bibliothek via USB 3.0 angesteuert und wird mithilfe eines Schachbrettmusters nach [164] bzgl. des Roboters kalibriert. Darüber hinaus werden die von `Zivid` bereitgestellten Rausch- und Ausreißerunterdrückungsverfahren eingesetzt. Aufgrund der Selbstkalibrierung der Kamera ist keine Kalibrie-

rung der Kameraparameter notwendig. Für die Aufnahme eines Tiefenbildes werden mehrere Aperturgrößen verwendet, die automatisch ermittelt werden. Dies verlangsamt zwar die Regelung, aber verbessert dafür die Schätzung des Tiefenbildes. Die Auflösung der Kamera beträgt 1.920×1.200 Pixel sowohl für das RGB- als auch für das Tiefenbild. Als Anker wird ein Anker eines Anlassermotors von `Bosch` verwendet. Um die Reflexionen des Kameralichtes zu vermindern, wird der Anker auf ein Material mit diffusen Reflexionseigenschaften gelegt. Der Gesamtaufbau und eine durch die `Zivid` aufgenommene Punktwolke des verwendeten Ankers sind in Abb. 6.10 illustriert.



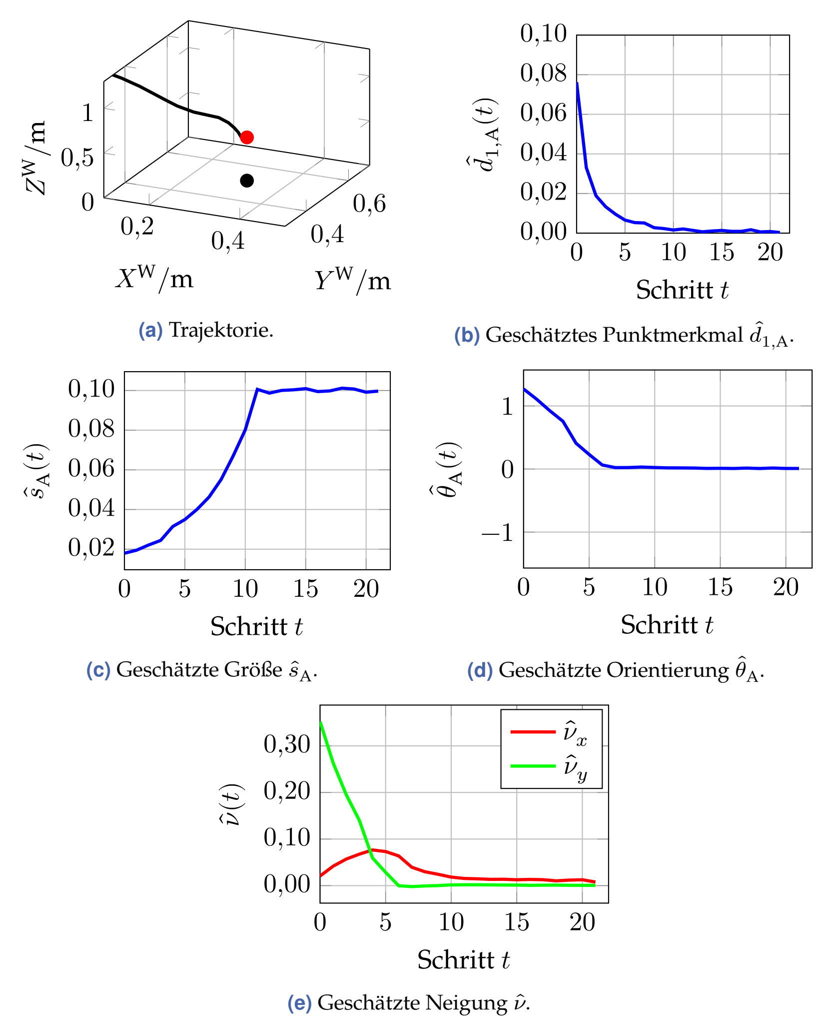
(a) Gesamtaufbau



(b) Punktwolke des verwendeten Ankers

Abbildung 6.10 Der Gesamtaufbau des Versuchs (links) besteht aus einem `UR10`-Roboter mit einer `Zivid`-Kamera, der auf einem Tisch mit metallischer Oberfläche der Größe 1 m $\times$ 1 m liegt. Der verwendete Anker (rechts) stammt aus einem Anlassermotor von `Bosch`. Das Bild zeigt eine aus dem Tiefenbild rekonstruierte Punktwolke des Ankers.

Der Regler verwendet die Neigung der Tischplatte für die Bestimmung von $[\omega_x, \omega_y]^T$, da sich dieses Merkmal als robuster als das Kantenmerkmal – insbesondere bei der gegebenen Ankerform – herausgestellt hat und die Symmetrieachse des Ankers parallel zur Tischebene ist. Außerdem wird das Ankermodell mit digitalem Zoom (VMZ) verwendet. Für die Eingabe in das U-Net wird das Bild auf ein Quadrat geschnitten und auf 336×336 Pixel heruntergetastet. Die geforderte Größe s_A^* des Ankers wird ggf. adaptiv verkleinert, um den Mindestabstand der `Zivid`-Kamera zum Objekt von 30 cm einzuhalten. Ansonsten gilt für die gewünschte Zielgröße des Ankers im Bild $s_A^* = 0{,}10$. Um nach der Regelung eine möglichst gute Trajektorie zu erhalten, sind möglichst geringere Fehlertoleranzen erforderlich.



(a) Trajektorie.

(b) Geschätztes Punktmerkmal $\hat{d}_{1,\mathrm{A}}$.

(c) Geschätzte Größe $\hat{s}_{\mathrm{A}}$.

(d) Geschätzte Orientierung $\hat{\theta}_{\mathrm{A}}$.

(e) Geschätzte Neigung $\hat{\nu}$.

Abbildung 6.11 Trajektorie und geschätzte Merkmale des realen Systems.

Da in der Punktwolke trotz Verwendung der Ausreißerunterdrückung dennoch Ausreißer und Punkte ohne Tiefeninformationen auftreten, wird für die Tiefenschätzung nicht wie in der Simulation die Tiefe des Mittelpunkts der geschätzten Ankermaske verwendet, sondern der Me-

dianwert der Tiefenkarte der `Zivid`-Kamera über ein Fenster der Größe 21×21. Darüber hinaus wird die Differenz aus geschätzter Tischebene und Tiefenkarte dazu verwendet, um die Segmentierung zu verbessern.

Die Ergebnisse sind in Abb. 6.11 dargestellt. Hierbei wird der Abstand $\hat{d}_{1,\mathrm{A}} = \sqrt{x_{1,\mathrm{A}}^2(t) + y_{1,\mathrm{A}}^2(t)}$ der Koordinaten des geschätzten Punktmerkmals $[\hat{x}_{1,\mathrm{A}}, \hat{y}_{1,\mathrm{A}}]^{\mathrm{T}}$ zum Ursprung dargestellt. Es ist ein leichtes Rauschen bei der geschätzten Position und Größe des Ankers im Bild feststellbar. Dieses ist auf kleine Fehler bei der Segmentierung durch Schatten, eine nicht festziehbare Halterung der Kamera am Endeffektor und Fehler infolge von Reflexionen des Streifenlichtes zurückzuführen. Die Merkmale konvergieren innerhalb von zehn Schritten gegen ihren jeweiligen Zielwert. Aufgrund des Rauschens sind zwölf weitere Schritte notwendig, um in den Toleranzbereich der Merkmalsfehler zu gelangen. Insgesamt ist aber eine stabile Regelung möglich.

6.3 Fazit

In diesem Kapitel werden das U-Net für die Segmentierung von Ankern und ein darauf aufbauender bildbasierten Regler beschrieben. Da nur eine geringe Anzahl an Lerndaten vorhanden ist, wird das U-Net beim Training mit stark augmentierten Daten trainiert, die sowohl klassische Bildverarbeitungsoperationen als auch Methoden der Domänenrandomisierung enthalten. Trotz möglicher logischer Widersprüche bei der vorliegenden Segmentierungsaufgabe liefert eine sigmoide Aktivierung der letzten Schicht die besten Resultate bzgl. der geschätzten Masken der Segmentierung. Das Konkatenieren von Kanteninformationen verbessert den gSDK leicht. Varianzuntersuchungen zeigen, dass das U-Net sehr robust gegenüber Variationen von Helligkeit, Orientierung, Position und Kontrast ist. Die Robustheit geht über die durch Augmentierung gezeigten Variationen hinaus. Letzteres gilt nicht für die Skalierung, da ein Anker nur im durch die Augmentierung gegebenen Bereich vollständig erkannt wird. Außerhalb dieses Bereichs ist aber die Ankermaske größtenteils erkennbar.

Um gegenüber diesen Degradierungen sowie fluktuativen Konstellationen robuster zu werden, wird die Ausgabe durch einen digitalen Zoom

und die Verwendung eines Modells stabilisiert. Ferner führt das Modell beim Auftreten mehrerer Objekte im Bild nicht zu einer Divergenz.

Der bildbasierte Regler bildet mit dem verarbeiteten Segmentierungsergebnis einen Gradienten, mit dem eine Kamera in gewünschter Weise zu einem Objekt ausgerichtet werden kann. Der Regler ist robust gegenüber Rauschen in Bezug auf die detektierten Merkmale und die Parameter der Kameramatrix. Bspw. kann ein suboptimales Kantenmerkmal statt der Schätzung der Bodenebene auch für die Regelung verwendet werden. Trotz auftretender Schwierigkeiten wie Reflexionen und Schatten im realen System zeigt der Regler auch hier ein robustes Verhalten.

Ausgehend von der gefundenen Position kann eine Trajektorie – im Beispiel des Ankers ist dies ein Halbkreis – ermittelt werden, um das Objekt abzuscannen und nach Irregularitäten auf der Oberfläche zu suchen.

7 Zusammenfassung

Im Rahmen dieser Arbeit wurde der Einsatz von konvolutionären neuronalen Netzen in der industriellen Bildverarbeitung betrachtet. Die vorliegende Arbeit unterteilt sich in den Entwurf effizienter neuronaler Netze für Klassifikationsaufgaben und die Entwicklung eines bildbasierten Reglers, bei dem klassische Bildverarbeitung mit Methoden des maschninellen Lernens kombiniert werden.

7.1 Effiziente CNNs

Hinsichtlich Genauigkeit und Rechenaufwand effiziente CNNs erlauben eine ökonomisch sinnvolle Implementierung auf mobilen Endgeräten oder FPGAs. Bezüglich der Genauigkeit wurde zunächst der Einfluss der Chargengröße beim Training mit Augmentierung untersucht. Die Ergebnisse deuten darauf hin, dass bei einer hinreichend guten Augmentierungsstrategie die erzielte Genauigkeit durch eine größere Chargengröße verbessert wird, wenn die Anzahl an Iterationen pro Durchlauf konstant ist. Dies ist mit der genaueren konsistenten Schätzung des Gradienten verbunden.

Ebenso konnte die Genauigkeit durch die Kombination von Vorverarbeitung und Augmentierung verbessert werden. Hierzu wurden die Vorverarbeitung bzw. Bildverbesserung als normalisierender und die Augmentierung mit einem abgewandelten *faster-AutoAugment*-Verfahren als integrativer Weg zum Erstellen invarianter Merkmale nach Schulz-Mirbach [137] interpretiert. Für die Vorverarbeitung wurde das aSVST-Verfahren entwickelt, bei dem datensatzspezifisch und gradientenbasiert nach der besten Kombination verschiedener Bildverbesserungsverfahren gesucht wird. Anhand der Beispieldatensätze wurde gezeigt, dass vorhandene irrelevante Informationen wie bspw. Farbe bei einem Ziffendatensatz durch aSVST stark reduziert werden, während relevante

Informationen wie Konturen und Risse bei Stahloberflächen verstärkt werden. Infolgedessen konnte eine Verbesserung der Genauigkeit erzielt werden. Auch wenn der Gewinn an Genauigkeit durch Augmentierung höher ist als durch aSVST, hat sich letzteres bei Redundanzanalysen als ressourcensparender erwiesen. Die Kombination beider Verfahren hat gezeigt, dass sich beide komplementär in der Auswahl ihrer Operationen verhalten und somit die resultierenden CNNs gegenüber einer höheren Anzahl an Merkmalen invariant sind, was ebenfalls zu einer höheren Genauigkeit führte. Ferner war hierdurch eine höhere Resilienz gegenüber Angriffen auf den Klassifikator feststellbar.

Durch die Verwendung der in dieser Arbeit vorgestellten statischen 8-Bit-Festkomma-Quantisierung ist auf entsprechender Hardware eine erhebliche Reduktion von Rechenkapazitäten möglich, welche mit einem geringen Verlust an Genauigkeit erkauft wird. Das Verfahren basiert auf dem Skalieren von Gewichten mit Skalierungsfaktoren, die in einem Vortrainingsschritt bestimmt werden müssen. Es wurde anhand von Histogrammen gezeigt, dass das Verfahren gewisse Anforderungen an die Breite des CNN stellt. Die Quantisierung wurde außerdem mit kürzeren Bitlängen erprobt, wobei hier der Verlust an Genauigkeit eine Feinabstimmung der Gewichte erfordert.

Die vorangegangenen Untersuchungen hinsichtlich der Quantisierung und der Eigenschaften der Datensätze haben gezeigt, dass eine optimale Topologie von zwei Faktoren abhängt: Während die Kapazität eines CNNs, d. h. die Tiefe und Breite, von der Komplexität des Lerndatensatzes abhängt, deutet das Quantisierungsverfahren auf eher schmale und tiefe CNNs hin. Daher wurde im Rahmen dieser Arbeit ein metaheuristisches Verfahren entwickelt, das in die Verarbeitungskaskade aus Vorverarbeitung, Augmentierung und Quantisierung eingebettet wurde. Mit der erzielten Genauigkeit als Fitnessfunktion und der Anzahl an MAC-Operationen und Gewichten als Strafterme wurde dabei nach Topologien gesucht, die auf einem handelsüblichen FPGA implementiert werden können. Im Ergebnis wurden für die verschiedenen Datensätze die prognostizierten tiefen, schmalen CNNs gefunden.

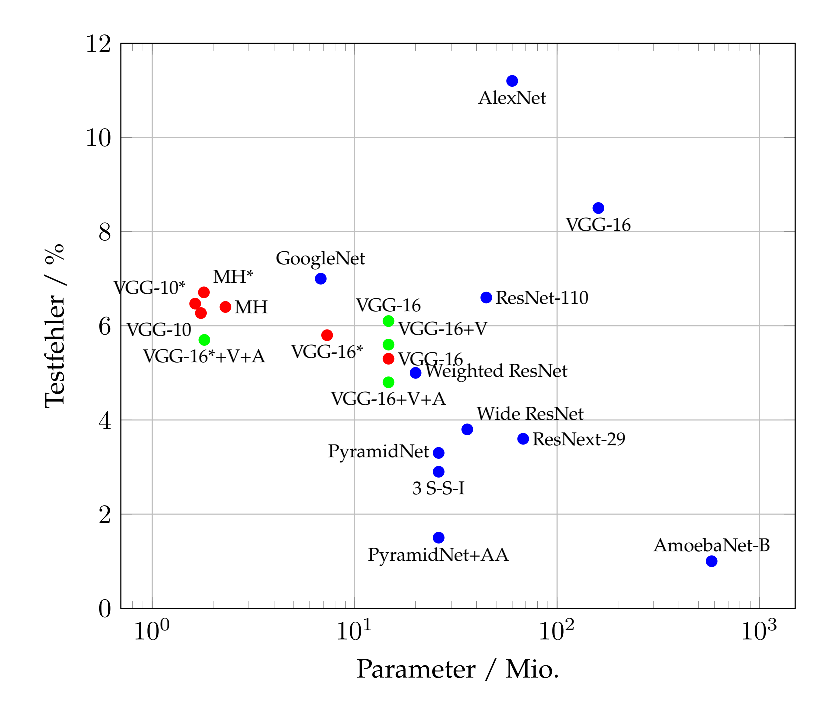


Abbildung 7.1 Erzielter Testfehler auf dem CIFAR-10-Datensatz abhängig von der Anzahl der Parameter für verschiedene CNN-Topologien [30, 48, 63, 92, 104, 127, 131, 139, 145, 155, 172] und eigene Verfahren mit (rot) und ohne (grün) Quantisierung. Bei nicht quantisierten CNNs bezeichnet +V die Verwendung von aSVST und +A die Verwendung von FAARO. Quantisierte Netze verwenden beide Verfahren. Mit * werden CNNs bezeichnet, auf die *Channel-Pruning* angewendet wurde. Das in dieser Arbeit verwendete VGG-16-Netz hat aufgrund des Weglassens der vollständig verbundenen Schichten weniger Parameter als das Original. Mit MH ist das Ergebnis der Metaheuristik gekennzeichnet.

Die sich in den einzelnen Verfahren dieser Arbeit ergebenen Topologien sind in Abb. 7.1 in die Grafik aus Kapitel 1 eingefügt. Einerseits wird durch aSVST-Vorverarbeitung und FAARO-Augmentierung der Testfehler verringert und andererseits wird durch *Channel-Pruning*, die Metaheuristik oder eine kleinere Topologie die Anzahl an Parametern um eine Größenordnung reduziert. Durch die Quantisierung und *Channel-Pruning* steigt jedoch der Fehler geringfügig an. Insgesamt ergibt sich eine neue Klasse an ressourcensparenden, aber dafür relativ performanten CNN-Topologien, die teilweise bessere Genauigkeiten als *GoogleNet*

oder *ResNet-110* für CIFAR-10 liefern bei deutlich geringerem Speicherbedarf der Inferenz. Dies ermöglicht die Implementierung der Inferenz auf den rund 7 MB des Block-RAM eines `Xilinx7`-FPGAs. Für industrielle Datensätze konnten ähnliche Resultate erzielt werden.

7.2 Robotergestützte Inspektion

Im zweiten Teil dieser Arbeit wurde ein Verfahren vorgestellt, das eine bildbasierte Regelung eines Roboterarms mit einer Tiefenbildkamera an dessen Endeffektor erlaubt. Das Verfahren wurde für Anker von Elektromotoren entworfen. Hierzu wurde ein Datensatz von verschiedenartigen Ankern erstellt, bei dem für den Anker und seine relevanten Komponenten Masken als Grundwahrheit für eine semantische Segmentierung erstellt wurden. Mithilfe dieser Daten wurde ein U-Net trainiert, wobei verschiedene Erweiterungen des U-Net und Augmentierungsstrategien analysiert wurden, um den negativen Auswirkungen aufgrund der geringen Anzahl an Bildern des Datensatzes entgegenzuwirken. Um die Ausgabe zu stabilisieren, wurde ein adaptives Ankermodell entwickelt und ein digitaler Zoom realisiert, der die Tatsache ausnutzt, dass der Anker bei einer ungünstigen Größe im Bild nur rudimentär erkannt wird.

Die Ausgabe des U-Net dient dem bildbasierten Regler, um die Lageparameter des Ankers zu schätzen, auf dessen Grundlage ein Regelschritt berechnet wird. Diese Parameter sind die Position im Bild, die relative Größe und die Orientierung. Um die Winkel zwischen optischer Achse der Kamera und der Rotationsachse des Ankers auszuregeln, hat sich die Schätzung der relativen Neigung der Bodenebene als vorteilhaft erwiesen. Diese Parameterwahl erlaubte es, einen entkoppelten und damit stabilen Regler zu entwerfen, der simulativ und in einer realen Umgebung evaluiert wurde.

7.3 Fazit und Ausblick

Maschinelles Lernen und insbesondere konvolutionäre neuronale Netze sind aufgrund ihrer hohen Performanz aus der heutigen Bildverarbeitung nicht mehr wegzudenken. Jedoch ist der Einsatz von CNNs in

sicherheitskritischen Bereichen und im industriellen Umfeld noch umstritten, da der *Black-Box*-Charakter von CNNs die Interpretation ihres Klassifikationsverhaltens erschwert und bereits relativ einfache Angriffe ihre Klassifikationsleistung erheblich schmälern können. Im industriellen Umfeld treten zusätzlich Bedenken zutage, dass vertrauliche Details über die Produktion an die entsprechenden Soft- bzw. Hardwareanbieter gelangen könnten.

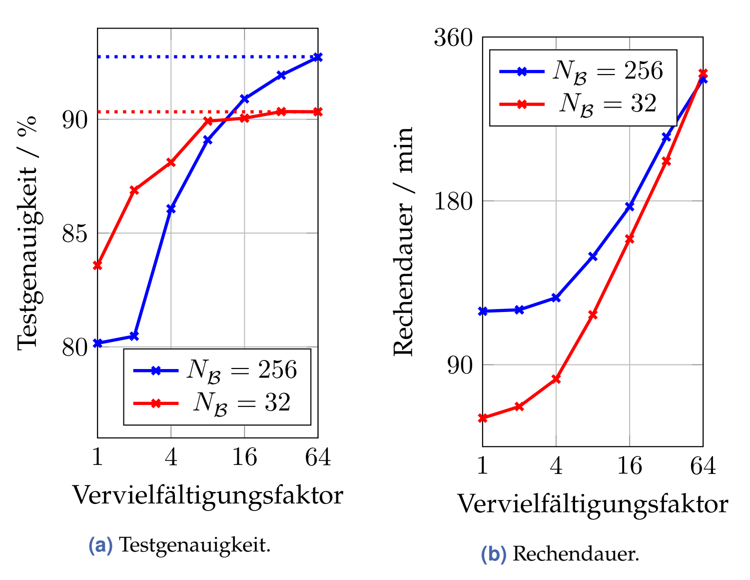
Im ersten Teil dieser Arbeit wurde nicht nur ein Framework für den Entwurf von CNNs für mobile Systeme bzw. FPGAs insbesondere für industrielle Anwendungen vorgestellt, sondern auch untersucht wie sich die Verarbeitung der Trainingsdaten sowohl positiv auf die Genauigkeit als auch auf die Resilienz gegenüber Angriffen auswirkt. Mithilfe der Interpretation von Vorverarbeitung und Augmentierung als Konstruktion invarianter Merkmale kann dieses Verhalten teilweise erklärt werden. Auch wenn hierdurch nicht alle Unklarheiten neuronaler Netze beseitigt werden konnten, kann die Interpretierbarkeit der Trainingsdaten schon zu einer höheren Akzeptanz in den oben genannten Bereichen führen, da ausgeschlossen werden kann, dass ein CNN aufgrund bestimmter Details wie z. B. Wasserzeichen in einzelnen Bildern, die auswendig gelernt wurden, eine Entscheidung trifft.

Die Kombination von konventioneller Bildverarbeitung mit neuronalen Netzen, wie sie im zweiten Teil der vorliegenden Arbeit beschrieben wird, birgt ein riesiges Potential: Einerseits, da Fehler von CNNs durch Plausibilitätsbetrachtungen reduzierbar sind und andererseits aufgrund der besseren Interpretierbarkeit durch die konventionellen Anteile. Hier eröffnen sich neue Forschungsgebiete. Bspw. können klassische Verfahren für Detektionsaufgaben, die auf einfachen Modellen basieren, durch die Fähigkeiten von neuronalen Netzen, auf Grundlage der Statistiken der Lerndaten optimale Entscheidungen zu liefern, verbessert werden. Hierbei sind nicht nur sequentielle Verarbeitungsketten möglich, sondern auch parallele Subprozesse, die Daten, Merkmale und Entscheidungen in Zwischenschritten miteinander fusionieren.

Anhang

A Anhang

A.1 Vergleich Online- und Offline-Augmentierung



(a) Testgenauigkeit.

(b) Rechendauer.

Abbildung A.1 Testgenauigkeit und Rechendauer für die Chargengrößen 32 und 256 bei Offline-Augmentierung im Vergleich zu Online-Augmentierung. Die gestrichelte Linie entspricht der Testgenauigkeit bei Online-Augmentierung.

Abb. A.1 zeigt die Testgenauigkeit und die Rechendauer beim Training mit 100 Durchläufen des CIFAR-10-Datensatzes auf einer `GeForce 2080 TI` mit einer Offline-Augmentierung mit den Operationen nach [30]. Bei einer Offline-Augmentierung wird der Datensatz vor dem Training mit den entsprechenden Operationen vervielfältigt, während bei der Online-Augmentierung die Bilder während des Trainings stochas-

tisch verarbeitet werden. Die Anzahl der Iteration pro Durchlauf ist fest, d. h. unabhängig von $N_{\mathcal{B}}$ (vgl. Abschnitt 3.1 und 4.2).

Während die Rechendauer mit der Größe des Datensatzes steigt, verlaufen die Kurven der Testgenauigkeit wie Sättigungskurven, die sich der jeweiligen Genauigkeit der Online-Augmentierung anpassen. Bei niedrigeren Chargengrößen tendiert die Genauigkeit, früher in Sättigung zu gehen. Bei $N_{\mathcal{B}} = 32$ reicht ein Vervielfältigungsfaktor von 8 bereits aus, wohingegen bei $N_{\mathcal{B}} = 256$ der Faktor 64 notwendig ist, welcher nicht nur einen enormen Speicherplatz benötigt, sondern auch die Rechendauer extrem erhöht.

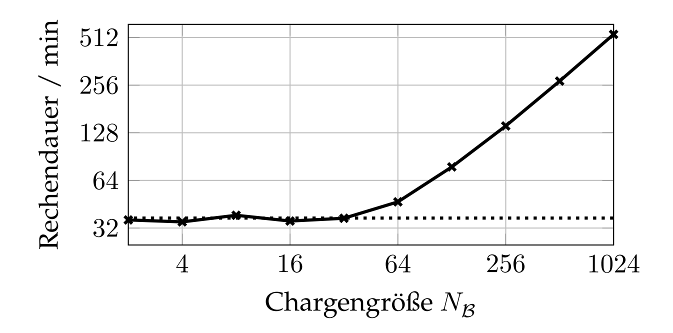
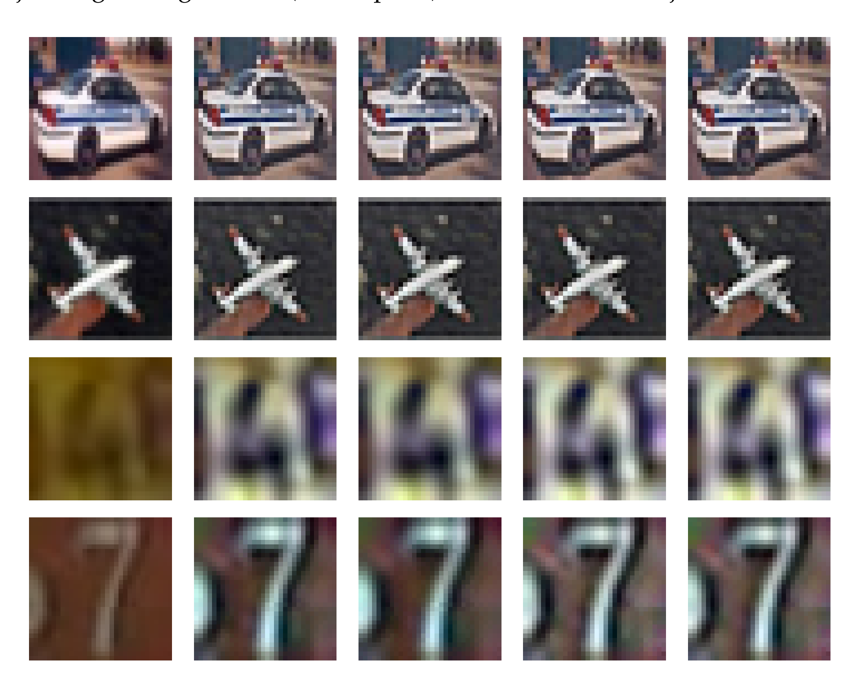


Abbildung A.2 Rechendauer abhängig von der Chargengröße bei Online-Augmentierung. Die gestrichelte Linie entspricht der theoretischen Rechendauer, die für eine GPU ohne Speicherlimitierung und eine parallelisierte Augmentierung auf der CPU erreicht wird.

Abb. A.2 zeigt, dass die Rechendauer bei Online-Augmentierung für $N_{\mathcal{B}} \leq 32$ unabhängig von der Chargengröße ist. Für $N_{\mathcal{B}} > 32$ steigt die Rechendauer linear im doppelt-logarithmischen Graphen an. Hier ist die Augmentierung und die Verarbeitung der Bilder zu Chargen, die beide auf der CPU ausgeführt werden, der Flaschenhals im Training.

A.2 Beispiele des aSV- und aSVST-Verfahrens

Im Folgenden werden für jeden der vier untersuchten Datensätze jeweils zwei Beispiele für die Varianten des Vorverarbeitungsverfahren aus Abschnitt 3.2.2 sowie deren Approximationen gezeigt. Rein optisch unterscheiden sich die vorverarbeiteten Bilder kaum. Verglichen mit dem jeweiligen Originalbild (linke Spalte) ist der Unterschied jedoch deutlich.



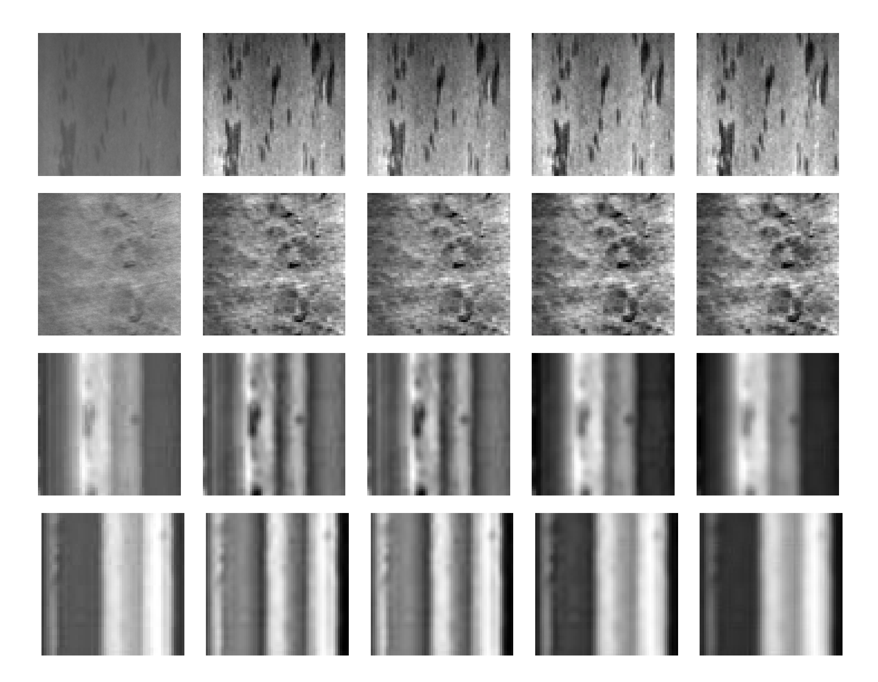


Abbildung A.3 Beispiele für die vorverarbeiteten Bilder nach dem aSV- bzw. aSVST-Verfahren mit und ohne Approximierung. Von oben nach unten sind jeweils zwei Beispiele der Datensätze CIFAR-10, SVHN, NEU und RSDDs. Jede Zeile zeigt (von links nach rechts) das Originalbild, das Ergebnis des aSV-Verfahrens, das Ergebnis des approximierten aSV-Verfahrens, das Ergebnis des aSVST-Verfahrens und das Ergebnis des approximierten aSVST-Verfahrens.

Literaturverzeichnis

[1] **Allibert, G., Courtial, E. und Chaumette, F.** *Predictive Control for Constrained Image-Based Visual Servoing*. In: *IEEE Transactions on Robotics* 26.5 (2010), S. 933–939.

[2] **Alom, Z., Josue, T., Rahman, N., Mitchell, W., Yakopcic, C. und Taha, T.** *Deep Versus Wide Convolutional Neural Networks for Object Recognition on Neuromorphic System*. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, S. 1–8.

[3] **Aristidou, A., Lasenby, J., Chrysanthou, Y. und Shamir, A.** *Inverse Kinematics Techniques in Computer Graphics: A Survey*. In: *Computer Graphics Forum*. Bd. 37. 6. Wiley Online Library. 2018, S. 35–58.

[4] **Azulay, A. und Weiss, Y.** *Why Do Deep Convolutional Networks Generalize So Poorly to Small Image Transformations?* In: *arXiv preprint arXiv:1805.12177* (2018).

[5] **Badrinarayanan, V., Kendall, A. und Cipolla, R.** *Segnet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), S. 2481–2495.

[6] **Bateux, Q., Marchand, E., Leitner, J., Chaumette, F. und Corke, P.** *Training Deep Neural Networks for Visual Servoing*. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, S. 3307–3314.

[7] **Bäuerle, S., Böhland, M., Barth, J., Reischl, M., Steimer, A. und Mikut, R.** *CAD-to-Real: Enabling Deep Neural Networks for 3D Pose Estimation of Electronic Control Units*. In: *at-Automatisierungstechnik* 69.10 (2021), S. 1–11.

[8] **Bengio, Y., Léonard, N. und Courville, A.** *Estimating or Propagating Gradients through Stochastic Neurons for Conditional Computation*. In: *arXiv preprint arXiv:1308.3432* (2013).

[9] **Bergmann, P., Löwe, S., Fauser, M., Sattlegger, D. und Steger, C.** *Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders*. In: *CoRR* abs/1807.02011 (2018). arXiv: 1807.02011.

[10] **Beyerer, J., Heizmann, M. und Kuntze, H.-B.** *Visual Servoing*. In: *at-Automatisierungstechnik* 60.5 (2012), S. 243–245.

[11] **Beyerer, J., Puente León, F. und Frese, C.** *Automatische Sichtprüfung – Grundlagen, Methoden und Praxis der Bildgewinnung und Bildauswertung*. Springer, 2012, S. I–XXIII, 1–940.

[12] **Bottema, O. und Roth, B.** *Theoretical Kinematics*. Bd. 24. Courier Corporation, 1990.

[13] **Bunkhumpornpat, C., Sinapiromsaran, K. und Lursinsap, C.** *DBSMOTE: Density-Based Synthetic Minority Over-Sampling Technique*. In: *Applied Intelligence* 36.3 (2012), S. 664–684.

[14] **Cai, Yang**. *How Many Pixels Do We Need to See Things?* In: *International Conference on Computational Science*. Springer. 2003, S. 1064–1073.

[15] **Calderon, S., Fallas, F., Zumbado, M., Tyrrell, P., Stark, H., Emersic, Z., Meden, B. und Solis, M.** *Assessing the Impact of the Deceived non Local Means Filter as a Preprocessing Stage in a Convolutional Neural Network Based Approach for Age Estimation Using Digital Hand X-Ray Images*. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, S. 1752–1756.

[16] **Chaumette, F.** *Potential Problems of Stability and Convergence in Image-Based and Position-Based Visual Servoing*. In: *The Confluence of Vision and Control*. Springer, 1998, S. 66–78.

[17] **Chaumette, F. und Hutchinson, S.** *Visual Servo Control. I. Basic Approaches*. In: *IEEE Robotics & Automation Magazine* 13.4 (2006), S. 82–90.

[18] **Chaumette, F. und Hutchinson, S.** *Visual Servo Control. II. Advanced Approaches [Tutorial]*. In: *IEEE Robotics & Automation Magazine* 14.1 (2007), S. 109–118.

[19] **Chawla, N. V., Bowyer, K. W., Hall, L. O. und Kegelmeyer, W. P.** *SMOTE: Synthetic Minority Over-Sampling Technique*. In: *Journal of Artificial Intelligence Research* 16 (2002), S. 321–357.

[20] **Chen, Q., Xin, C., Zou, C., Wang, X. und Wang, B.** *A Low Bit-Width Parameter Representation Method for Hardware-Oriented Convolution Neural Networks*. In: *IEEE 12th International Conference on ASIC (ASICON) 2017*. IEEE. 2017, S. 148–151.

[21] **Chen, T., Fu, G. und Wang H.and Li, Y.** *Research on Influence of Image Preprocessing on Handwritten Number Recognition Accuracy*. In: *The 8th International Conference on Computer Engineering and Networks (CENet2018)*. Cham: Springer International Publishing, 2020, S. 253–260.

[22] **Cheng, Y. und Yan J.and Wang, Z.** *Enhancement of Weakly Illuminated Images by Deep Fusion Networks*. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, S. 924–928.

[23] **Choi, Y., Kim, N., Hwang, S. und Kweon, I. So**. *Thermal Image Enhancement Using Convolutional Neural Network*. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, S. 223–230.

[24] **Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T. und Ronneberger, O.** *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, S. 424–432.

[25] **Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S. und Vedaldi, A.** *Describing Textures in the Wild*. In: *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[26] **Cobbe, K., Klimov, O., Hesse, C., Kim, T. und Schulman, J.** *Quantifying Generalization in Reinforcement Learning*. In: *Proceedings of the 36th International Conference on Machine Learning*. Hrsg. von **Tschaudhuri, K. und Salachutdinow, R.** Bd. 97. Proceedings of Machine Learning Research. PMLR, 2019, S. 1282–1289.

[27] **Corke, P. I. und Hutchinson, S. A.** *A New Partitioned Approach to Image-Based Visual Servo Control*. In: *IEEE Transactions on Robotics and Automation* 17.4 (2001), S. 507–515.

[28] **Courbariaux, M., Bengio, Y. und David, J.-P.** *Training Deep Neural Networks with Low Precision Multiplications*. In: *arXiv preprint arXiv:1412.7024* (2014).

[29] **Crum, W. R., Camara, O. und Hill, D. L. G.** *Generalized Overlap Measures for Evaluation and Validation in Medical Image Analysis*. In: *IEEE Transactions on Medical Imaging* 25.11 (2006), S. 1451–1461.

[30] **Cubuk, E., Zoph, B., Mané, D., Vasudevan, V. und Le, Q.** *AutoAugment: Learning Augmentation Policies from Data*. In: *CoRR* abs/1805.09501 (2018). arXiv: 1805.09501.

[31] **Cubuk, E., Zoph, B., Shlens, J. und Le, Q.** *Randaugment: Practical Automated Data Augmentation with a Reduced Search Space*. In: *Proceedings of the IEEECVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, S. 702–703.

[32] **Deng, J., Dong, W., Socher, R., Li, L., Li, K. und Fei-Fei, L.** *ImageNet: A Large-Scale Hierarchical Image Database*. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee. 2009, S. 248–255.

[33] **Denil, M., Shakibi, B., Dinh, L., Ranzato, M. und de Freitas, N.** *Predicting Parameters in Deep Learning*. In: *CoRR* abs/1306.0543 (2013). arXiv: 1306.0543.

[34] **DeVries, T. und Taylor, G. W.** *Dataset Augmentation in Feature Space*. In: *arXiv preprint arXiv:1702.05538* (2017).

[35] **DeVries, T. und Taylor, G. W.** *Improved Regularization of Convolutional Neural Networks with Cutout*. In: *CoRR* abs/1708.04552 (2017). arXiv: 1708.04552.

[36] **Diakogiannis, F. I., Waldner, F., Caccetta, P. und Wu, C.** *ResUNet-A: a Deep Learning Framework for Semantic Segmentation of Remotely Sensed Data*. In: *CoRR* abs/1904.00592 (2019). arXiv: 1904.00592.

[37] **Dodge, S. und Karam, L.** *Understanding How Image Quality Affects Deep Neural Networks*. In: *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2016, S. 1–6.

[38] **Dodge, S. und Karam, L.** *A Study and Comparison of Human and Deep Learning Recognition Performance under Visual Distortions*. In: *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2017, S. 1–7.

[39] **Du, K. und Swamy, M.** *Search and Optimization by Metaheuristics*. Springer, 2016.

[40] **Erhan, D., Bengio, Y., Courville, A. und Vincent, P.** *Visualizing Higher-Layer Features of a Deep Network*. In: *University of Montreal* 1341.3 (2009), S. 1.

[41] **Espiau, B.** *Effect of Camera Calibration Errors on Visual Servoing in Robotics*. In: *Experimental robotics III*. Springer, 1994, S. 182–192.

[42] **Espiau, B., Chaumette, F. und Rives, P.** *A New Approach to Visual Servoing in Robotics*. In: *IEEETransactions on Robotics and Automation* 8.3 (1992), S. 313–326.

[43] **Feddema, J. T. und Mitchell, O. R.** *Vision-Guided Servoing with Feature-Based Trajectory Generation (for Robots)*. In: *IEEE Transactions on Robotics and Automation* 5.5 (1989), S. 691–700.

[44] **Fernando, C., Banarse, D. Ian, Reynolds, M., Besse, F., Pfau, D., Jaderberg, M., Lanctot, M. und Wierstra, D.** *Convolution by Evolution: Differentiable Pattern Producing Networks*. In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. 2016, S. 109–116.

[45] **Fischler, M. A. und Bolles, R. C.** *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. In: *Communications of the ACM* 24.6 (1981), S. 381–395.

[46] **Floreano, D., Dürr, P. und Mattiussi, C.** *Neuroevolution: From Architectures to Learning*. In: *Evolutionary Intelligence* 1.1 (2008), S. 47–62.

[47] **Gan, J., Li, Q., Wang, J. und Yu, H.** *A Hierarchical Extractor-Based Visual Rail Surface Inspection System*. In: *IEEE Sensors Journal* 17 (2017), S. 7935–7944.

[48] **Gastaldi, X.** *Shake-Shake Regularization*. 2017. arXiv: 1705.07485 [cs.LG].

[49] **Goldenberg, A., Benhabib, B. und Fenton, R.** *A Complete Generalized Solution to the Inverse Kinematics of Robots*. In: *IEEE Journal on Robotics and Automation* 1.1 (1985), S. 14–20.

[50] **Golmant, N., Vemuri, N., Yao, Z., Feinberg, V., Gholami, A., Rothauge, K., Mahoney, M. W. und Gonzalez, J.** *On the Computational Inefficiency of Large Batch Sizes for Stochastic Gradient Descent*. In: *arXiv preprint arXiv:1811.12941* (2018).

[51] **Goodfellow, I., Bengio, Y. und Courville, A.** *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[52] **Goodfellow, I., Shlens, J. und Szegedy**. *Explaining and Harnessing Adversarial Examples*. In: *arXiv preprint arXiv:1412.6572* (2014).

[53] **Goodman, D., Xin, H., Yang, W., Yuesheng, W., Junfeng, X. und Huan, Z.** *Advbox: A Toolbox to Generate Adversarial Examples that Fool Neural Networks*. In: *arXiv preprint arXiv:2001.05574* (2020).

[54] **Goyal, P., Dollar, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y. und He, K.** *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*. In: *arXiv preprint arXiv:1706.02677* (2017).

[55] **Graham, B.** *Kaggle Diabetic Retinopathy Detection Competition Report*. In: *University of Warwick* (2015).

[56] **Guo, K., Sui, L., Qiu, J., Yao, S., Han, S., Wang, Y. und Yang, H.** *Angel-Eye: A Complete Design Flow for Mapping CNN onto Customized Hardware*. In: *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE. 2016, S. 24–29.

[57] **Gysel, P., Motamedi, M. und Ghiasi, S.** *Hardware-Oriented Approximation of Convolutional Neural Networks*. In: *arXiv preprint arXiv:1604.03168* (2016).

[58] **Han, D., Kim, J. und Kim, J.** *Deep Pyramidal Residual Networks*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, S. 5927–5935.

[59] **Han, S., Mao, H. und Dally, W. J.** *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. In: *arXiv preprint arXiv:1510.00149* (2015).

[60] **Han, S., Pool, J., Narang, S., Mao, H., Gong, E., Tang, S., Elsen, E., Vajda, P., Paluri, M. und Tran, J.** *DSD: Dense-Sparse-Dense Training for Deep Neural Networks*. In: *arXiv preprint arXiv:1607.04381* (2016).

[61] **Hataya, R., Zdenek, J., Yoshizoe, K. und Nakayama, H.** *Faster AutoAugment: Learning Augmentation Strategies Using Backpropagation*. In: *arXiv preprint arXiv:1911.06987* (2019).

[62] **Hauberg, S., Freifeld, O., Larsen, A., Fisher, J. und Hansen, L.** *Dreaming More Data: Class-Dependent Distributions over Diffeomorphisms for Learned Data Augmentation*. In: *Artificial Intelligence and Statistics*. 2016, S. 342–350.

[63] **He, K., Zhang, X., Ren, S. und Sun, J.** *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[64] **He, Y., Zhang, X. und Sun, J**. *Channel Pruning for Accelerating Very Deep Neural Networks*. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, S. 1389–1397.

[65] **Heckbert, P.** *Graphics Gems IV (IBM Version)*. Elsevier, 1994.

[66] **Hernández-Garcia, A. und König, P.** *Data Augmentation Instead of Explicit Regularization*. In: *arXiv preprint arXiv:1806.03852* (2018).

[67] **Hernández-Garcia, A., Mehrer, J., Kriegeskorte, N., König, P. und Kietzmann, T. C.** *Deep Neural Networks Trained with Heavier Data Augmentation Learn Features Closer Ro representations in hIT*. In: *Conference on Cognitive Computational Neuroscience*. 2018.

[68] **Hoffer, E., Hubara, I. und Soudry, D.** *Train Longer, Generalize Better: Closing the Generalization Gap in Large Batch Training of Neural Networks*. In: *arXiv preprint arXiv:1705.08741* (2017).

[69] **Holland, J.** *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.

[70] **Horowitz, M.** *Computing's Energy Problem (and What We Can Do About It)*. In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE. 2014, S. 10–14.

[71] **Hosoda, K. und Asada, M.** *Versatile Visual Servoing Without Knowledge of True Jacobian*. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*. Bd. 1. IEEE. 1994, S. 186–193.

[72] **Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. und Adam, H.** *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. In: *arXiv preprint arXiv:1704.04861* (2017).

[73] **Huang, S., Papernot, N., Goodfellow, I., Duan, Y. und Abbeel, P.** *Adversarial Attacks on Neural Network Policies*. In: *arXiv preprint arXiv:1702.02284* (2017).

[74] **Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, M., Chen, D., Lee, H., Ngiam, J., Le, Q. und Wu, Y.** *GPipe: Efficient Training of Giant Neural Networks Using Pipeline Parallelism*. In: *arXiv preprint arXiv:1811.06965* (2018).

[75] **Hutchinson, S., Hager, G. D. und Corke, P. I.** *A Tutorial on Visual Servo Control*. In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), S. 651–670.

[76] **Inoue, H.** *Data Augmentation by Pairing Samples for Images Classification*. In: *arXiv preprint arXiv:1801.02929* (2018).

[77] **Ioffe, S. und Szegedy, C.** *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. In: *International Conference on Machine Learning*. 2015, S. 448–456.

[78] **Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. und Kalenichenko, D**. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, S. 2704–2713.

[79] **Jalalvand, A., De Neve, W., Van de Walle, R. und Martens, J.** *Towards Using Reservoir Computing Networks for Noise-Robust Image Recognition*. eng. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. Vancouver, Canada: Institute of Electrical und Electronics Engineers (IEEE), 2016, S. 1666–1672.

[80] **Jang, E., Gu, S. und Poole, B.** *Categorical Reparameterization with Gumbel-Softmax*. In: *arXiv preprint arXiv:1611.01144* (2016).

[81] **Jang, W., Kim, K., Chung, M. und Bien, Z.** *Concepts of Augmented Image Space and Transformed Feature Space for Efficient Visual Servoing of an "Eye-in-Hand Robot"*. In: *Robotica* 9.2 (1991), S. 203–212.

[82] **Jo, J. und Bengio, Y.** *Measuring the Tendency of CNNs to Learn Surface Statistical Regularities*. In: *arXiv preprint arXiv:1711.11561* (2017).

[83] **Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J. und Aila, T.** *Training Generative Adversarial Networks with Limited Data*. In: *arXiv preprint arXiv:2006.06676* (2020).

[84] **Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. und Tang, P. T. P.** *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. In: *arXiv preprint arXiv:1609.04836* (2016).

[85] **Kingma, D. P. und Ba, J.** *Adam: A Method for Stochastic Optimization*. In: *arXiv preprint arXiv:1412.6980* (2015).

[86] **Kiryati, N., Eldar, Y. und Bruckstein, A. M.** *A Probabilistic Hough Transform*. In: *Pattern Recognition* 24.4 (1991), S. 303–316.

[87] **Koditschek, D. E.** *Robot Kinematics and Coordinate Transformations*. In: *1985 24th IEEE Conference on Decision and Control*. 1985, S. 1–4.

[88] **Koushik, J.** *Understanding Convolutional Neural Networks*. In: *arXiv preprint arXiv:1605.09081* (2016).

[89] **Krešo, I., Čaušević, D., Krapac, J. und Šegvić, S.** *Convolutional Scale Invariance for Semantic Segmentation*. In: *Pattern Recognition*. Hrsg. von **Rosenhahn, B. und Andres, B.** Cham: Springer International Publishing, 2016, S. 64–75.

[90] **Krizhevsky, A.** *One Weird Trick for Parallelizing Convolutional Neural Networks*. In: *arXiv preprint arXiv:1404.5997* (2014).

[91] **Krizhevsky, A. und Hinton, G.** *Learning Multiple Layers of Features from Tiny Images*. In: *University of Tront* (2009).

[92] **Krizhevsky, A., Sutskever, I. und Hinton, G. E.** *Imagenet Classification with Deep Convolutional Neural Networks*. In: *Advances in Neural Information Processing Systems*. 2012, S. 1097–1105.

[93] **Kubilius, J., Bracci, S. und Op de Beeck, H.** *Deep Neural Networks as a Computational Model for Human Shape Sensitivity*. In: *PLoS Computational Biology* 12.4 (2016), e1004896.

[94] **Kuo, R. und Zulvia, F.** *The Gradient Evolution Algorithm: A New Metaheuristic*. In: *Information Sciences* 316 (2015), S. 246–265.

[95] **Lee, A. X., Levine, S. und Abbeel, P.** *Learning Visual Servoing with Deep Features and Fitted Q-Iteration*. In: *arXiv preprint arXiv:1703.11000* (2017).

[96] **Lee, K., Lee, J., Lee, J., Hwang, S. und Lee, S.** *Brightness-Based Convolutional Neural Network for Thermal Image Enhancement*. In: *IEEE Access* 5 (2017), S. 26867–26879.

[97] **Leung, F., Lam, H., Ling, S. und Tam, P.** *Tuning of the Structure and Parameters of a Neural Network Using an Improved Genetic Algorithm*. In: *IEEE Transactions on Neural networks* 14.1 (2003), S. 79–88.

[98] **Li, B., Wu, F., Lim, S., Belongie, S. und Weinberger, K. Q.** *On Feature Normalization and Data Augmentation*. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, S. 12383–12392.

[99] **Li, C., Guo, C., Ren, W., Cong, R., Hou, J., Kwong, S. und Tao, D.** *An Underwater Image Enhancement Benchmark Dataset and Beyond*. In: *IEEE Transactions on Image Processing* (2019).

[100] **Li, H., Xu, Z., Taylor, G., Studer, C. und Goldstein, T.** *Visualizing the Loss Landscape of Neural Nets*. In: *arXiv preprint arXiv:1712.09913* (2017).

[101] **Li, Jiawei, Dai, Tao, Tang, Qingtao, Xing, Yeli und Xia, Shu-Tao**. *Cyclic Annealing Training Convolutional Neural Networks for Image Classification with Noisy Labels*. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, S. 21–25.

[102] **Lim, S., Kim, I., Kim, T., Kim, C. und Kim, S.** *Fast AutoAugment*. In: *arXiv preprint arXiv:1905.00397* (2019).

[103] **Lin, D., Talathi, S. und Annapureddy, S.** *Fixed Point Quantization of Deep Convolutional Networks*. In: *International Conference on Machine Learning*. 2016, S. 2849–2858.

[104] **Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B. und Belongie, S.** *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV].

[105] **Lin, T., Stich, S. U., Patel, K. K. und Jaggi, M.** *Don't Use Large Mini-Batches, Use Local SGD*. In: *arXiv preprint arXiv:1808.07217* (2018).

[106] **Lindeberg, T.** *Scale-Space Theory in Computer Vision*. Bd. 256. Springer Science & Business Media, 2013.

[107] **Liu, H., Simonyan, K. und Yang, Y.** *Darts: Differentiable Architecture Search*. In: *arXiv preprint arXiv:1806.09055* (2018).

[108] **Liu, Z., Afrinaldi, F., Zhang, H. und Jiang, Q.** *Exploring Optimal Timing for Remanufacturing Based on Replacement Theory*. In: *CIRP Annals* 65.1 (2016), S. 447–450.

[109] **Luo, W., Li, Y., Urtasun, R. und Zemel, R.** *Understanding the Effective Receptive Field in Deep Convolutional Neural Networks*. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, S. 4905–4913.

[110] **Manocha, D. und Canny, J. F.** *Efficient Inverse Kinematics for General 6R Manipulators*. In: *IEEE Transactions on Robotics and Automation* 10.5 (1994), S. 648–657.

[111] **Mareczek, J.** *Direkte Kinematik*. In: *Grundlagen der Roboter-Manipulatoren–Band 1*. Springer, 2020, S. 41–116.

[112] **Marr, D. und Hildreth, E.** *Theory of Edge Detection*. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 207.1167 (1980), S. 187–217.

[113] **Masters, D. und Luschi, C.** *Revisiting Small Batch Training for Deep Neural Networks*. In: *arXiv preprint arXiv:1804.07612* (2018).

[114] **Mathew, M., Desappan, K., Kumar Swami, P. und Nagori, S.** *Sparse, Quantized, Full Frame CNN for Low Power Embedded Devices*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2017.

[115] **McCulloch, W. S. und Pitts, W.** *A Logical Calculus of the Ideas Immanent in Nervous Activity*. In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), S. 115–133.

[116] **Miljković, Z., Mitić, M., Lazarević, M. und Babić, B.** *Neural Network Reinforcement Learning for Visual Control of Robot Manipulators*. In: *Expert Systems with Applications* 40.5 (2013), S. 1721–1736.

[117] **Mishkin, D., Sergievskiy, N. und Matas, J.** *Systematic Evaluation of Convolution Neural Network Advances on the ImageNet*. In: *Computer Vision and Image Understanding* 161 (2017), S. 11–19.

[118] **Montavon, G., Samek, W. und Müller, K.-R.** *Methods for Interpreting and Understanding Deep Neural Networks*. In: *Digital Signal Processing* 73 (2018), S. 1–15.

[119] **Moosavi-Dezfooli, S.-M., Fawzi, A. und Frossard, P.** *Deepfool: A Simple and Accurate Method to Fool Deep Neural Networks*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, S. 2574–2582.

[120] **Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. und Ng, A.** *Reading Digits in Natural Images with Unsupervised Feature Learning*. In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. Bd. 2011. 2. 2011, S. 5.

[121] **Noord, N. van und Postma, E.** *Learning Scale-Variant and Scale-Invariant Features for Deep Image Classification*. In: *Pattern Recognition* 61 (2017), S. 583–592.

[122] **Pal, K. und Sudeep, K.** *Preprocessing for Image Classification by Convolutional Neural Networks*. In: *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE. 2016, S. 1778–1781.

[123] **Pitaloka, D., Wulandari, A., Basaruddin, T. und Liliana, D.** *Enhancing CNN with Preprocessing Stage in Automatic Emotion Recognition*. In: *Procedia computer science* 116 (2017), S. 523–529.

[124] **Puang, E. Y., Tee, K. P. und Jing, W.** *KOVIS: Keypoint-based Visual Servoing with Zero-Shot Sim-to-Real Transfer for Robotics Manipulation*. In: *arXiv preprint arXiv:2007.13960* (2020).

[125] **Rachmadi, R. und Purnama, I.** *Vehicle Color Recognition Using Convolutional Neural Network*. In: *arXiv preprint arXiv:1510.07391* (2015).

[126] **Radiuk, P. M.** *Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets*. In: *Information Technology and Management Science* 20.1 (2017), S. 20–24.

[127] **Real, R., Aggarwal, A., Huang, A. und Le, Q.** *Regularized Evolution for Image Classifier Architecture Search*. 2019. arXiv: 1802.01548 [cs.NE].

[128] **Ronneberger, O., Fischer, P. und Brox, T.** *U-Net: Convolutional Networks for Biomedical Image Segmentation*. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, S. 234–241.

[129] **Rosenblatt, F.** *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. In: *Psychological Review* 65.6 (1958), S. 386.

[130] **Rublee, E., Rabaud, V., Konolige, K. und Bradski, G.** *ORB: An Efficient Alternative to SIFT or SURF*. In: *2011 International Conference on Computer Vision*. Ieee. 2011, S. 2564–2571.

[131] **Sagorujko, S. und Komodakis, N.** *Wide Residual Networks*. In: *arXiv preprint arXiv:1605.07146* (2016).

[132] **Saida, M., Hirata, Y. und Kosuge, K.** *Motion Control of Caster-Type Passive Mobile Robot with Servo Brakes*. In: *Advanced Robotics* 26.11-12 (2012), S. 1271–1290.

[133] **Salomon, R.** *Evolutionary Algorithms and Gradient Search: Similarities and Differences*. In: *IEEE Transactions on Evolutionary Computation* 2.2 (1998), S. 45–55.

[134] **Sampedro, C., Rodriguez-Ramos, A., Gil, I., Mejias, L. und Campoy, P.** *Image-Based Visual Servoing Controller for Multirotor Aerial Robots Using Deep Reinforcement Learning*. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, S. 979–986.

[135] **Saxena, A., Pandya, H., Kumar, G., Gaud, A. und Krishna, K. M.** *Exploring Convolutional Networks for End-to-End Visual Servoing*. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, S. 3817–3823.

[136] **Scherer, D., Müller, A. und Behnke, S.** *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition*. In: *International Conference on Artificial Neural Networks*. Springer. 2010, S. 92–101.

[137] **Schulz-Mirbach, H.** *Constructing Invariant Features by Averaging Techniques*. In: *12th IAPR International Conference on Pattern Recognition, Conference B: Patern Recognition and Neural Networks, ICPR 1994, Jerusalem, Israel, 9-13 October, 1994, Volume 2*. 1994, S. 387–390.

[138] **Sejnowski, T.** *The Deep Learning Revolution*. MIT Press, 2018.

[139] **Shen, F., Gan, R. und Zeng, G.** *Weighted Residuals for Very Deep Networks*. In: *2016 3rd International Conference on Systems and Informatics (ICSAI)*. IEEE. 2016, S. 936–941.

[140] **Sheng, T., Feng, C., Zhuo, S., Zhang, X., Shen, L. und Aleksic, M.** *A Quantization-Friendly Separable Convolution for MobileNets*. In: *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*. IEEE. 2018, S. 14–18.

[141] **Shorten, C. und Khoshgoftaar, T. M.** *A Survey on Image Data Augmentation for Deep Learning*. In: *Journal of Big Data* 6.1 (2019), S. 1–48.

[142] **Shrikumar, A., Greenside, P., Shcherbina, A. und Kundaje, A.** *Not just a Black Box: Learning Important Features through Propagating Activation Differences*. In: *arXiv preprint arXiv:1605.01713* (2016).

[143] **Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D. und Graepel, T.** *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. In: *CoRR* abs/1712.01815 (2017). arXiv: 1712.01815.

[144] **Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M. und Bolton, A.** *Mastering the Game of Go without Human Knowledge*. In: *Nature* 550.7676 (2017), S. 354–359.

[145] **Simonyan, K. und Zisserman, A.** *Very Deep Convolutional Networks for Large-Scale Image Recognition*. In: *arXiv preprint arXiv:1409.1556* (2014).

[146] **Singh, B. und Davis, L. S.** *An Analysis of Scale Invariance in Object Detection sSip*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, S. 3578–3587.

[147] **Song, K., Hu, S. und Yan, Y.** *Automatic Recognition of Surface Defects on Hot-Rolled Steel Strip Using Scattering Convolution Network*. In: *Journal of Computational Information Systems* 10.7 (2014), S. 3049–3055.

[148] **Sree Sharmila, T., Ramar, K. und Sree Renga Raja, T.** *Impact of Applying Pre-Processing Techniques for Improving Classification Accuracy*. In: *Signal, Image and Video Processing* 8.1 (2014), S. 149–157.

[149] **Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. und Salakhutdinov, R.** *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. In: *The Journal of Machine Learning Research* 15.1 (2014), S. 1929–1958.

[150] **Stanley, K. und Miikkulainen, R.** *Evolving Neural Networks Through Augmenting Topologies*. In: *Evolutionary Computation* 10.2 (2002), S. 99–127.

[151] **Strong, D. und Chan, T.** *Edge-Preserving and Scale-Dependent Properties of Total Variation Regularization*. In: *Inverse Problems* 19.6 (2003), S165.

[152] **Su, J., Vargas, D. Vasconcellos und Sakurai, K.** *One Pixel Attack for Fooling Deep Neural Networks*. In: *IEEE Transactions on Evolutionary Computation* 23.5 (2019), S. 828–841.

[153] **Sudre, C., Li, W., Vercauteren, T., Ourselin, S. und Jorge Cardoso, M.** *Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations*. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Cham: Springer International Publishing, 2017, S. 240–248.

[154] **Suganuma, M., Shirakawa, S. und Nagao, T.** *A Genetic Programming Approach to Designing Convolutional Neural Network Architectures*. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2017, S. 497–504.

[155] **Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Van Houcke, V. und Rabinovich, A.** *Going Deeper with Convolutions*. In: *Computer Vision and Pattern Recognition (CVPR)*. 2015.

[156] **Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. und Wojna, Z.** *Rethinking the Inception Architecture for Computer Vision*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, S. 2818–2826.

[157] **Tachibana, Y., Obata, T., Kershaw, J., Sakaki, H., Urushihata, T., Omatsu, T., Kishimoto, R. und Higashi, T.** *The Utility of Applying Various Image Preprocessing Strategies to Reduce the Ambiguity in Deep Learning-based Clinical Image Diagnosis*. In: *Magnetic Resonance in Medical Sciences* advpub (2019).

[158] **Talebi, H. und Milanfar, P.** *Learned Perceptual Image Enhancement*. In: *2018 IEEE International Conference on Computational Photography (ICCP)*. IEEE. 2018, S. 1–13.

[159] **Tamura, Shinichi, Kawai, Hideo und Mitsumoto, Hiroshi**. *Male/Female Identification from 8× 6 Very Low Resolution Face Images by Neural Network*. In: *Pattern recognition* 29.2 (1996), S. 331–335.

[160] **Tokuda, F., Arai, S. und Kosuge, K.** *Convolutional Neural Network-Based Visual Servoing for Eye-to-Hand Manipulator*. In: *IEEE Access* 9 (2021), S. 91820–91835.

[161] **Tolio, T., Bernard, A., Colledani, M., Kara, S., Seliger, G., Duflou, J., Battaia, O. und Takata, S.** *Design, Management and Control of Demanufacturing and Remanufacturing Systems*. In: *CIRP Annals* 66.2 (2017), S. 585–609.

[162] **Tran, T., Pham, T., Carneiro, G., Palmer, L. und Reid, I.** *A Bayesian Data Augmentation Approach for Learning Deep Models*. In: *Advances in Neural Information Processing Systems*. Hrsg. von **Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. und Garnett, R.** Bd. 30. Curran Associates, Inc., 2017.

[163] **Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S. und Birchfield, S.** *Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, S. 969–977.

[164] **Tsai, R. Y. und Lenz, RK**. *Real Time Versatile Robotics Hand/Eye Calibration Using 3D Machine Vision*. In: *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. IEEE. 1988, S. 554–561.

[165] **Wang, J. und Perez, L.** *The Effectiveness of Data Augmentation in Image Classification Using Deep Learning*. In: *Convolutional Neural Networks Vis. Recognit* (2017).

[166] **Wang, Y., Wang, H., Liu, Z. und Chen, W.** *Visual Servo-Collision Avoidance Hybrid Task by Considering Detection and Localization of Contact for a Soft Manipulator*. In: *IEEE/ASME Transactions on Mechatronics* 25.3 (2020), S. 1310–1321.

[167] **Weiss, L., Sanderson, A. und Neuman, C.** *Dynamic Sensor-Based Control of Robots with Visual Feedback*. In: *IEEE Journal on Robotics and Automation* 3.5 (1987), S. 404–417.

[168] **Weiss, L. E., Sanderson, A. C. und Neuman, C. P.** *Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach*. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Bd. 2. IEEE. 1985, S. 662–668.

[169] **Wong, S. C., Gatt, A., Stamatescu, V. und McDonnell, M. D.** *Understanding Data Augmentation for Classification: When to Warp?* In: *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE. 2016, S. 1–6.

[170] **Wurster, M., Häfner, B., Gauder, D., Stricker, N. und Lanza, G.** *Fluid Automation - A Definition and an Application in Remanufacturing Production Systems*. In: *Procedia CIRP* 97 (2021), S. 508–513.

[171] **Xiao, H., Rasul, K. und Vollgraf, R.** *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. In: *arXiv preprint arXiv:1708.07747* (2017).

[172] **Xie, L. und Yuille, A.** *Genetic CNN*. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, S. 1379–1388.

[173] **Xu, D., Lu, J., Wang, P., Zhang, Z. und Liang, Z.** *Partially Decoupled Image-Based Visual Servoing Using Different Sensitive Features*. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.8 (2017), S. 2233–2243.

[174] **Yamazaki, S., Mochimaru, M. und Kanade, T.** *Simultaneous Self-Calibration of a Projector and a Camera Using Structured Light*. In: *CVPR 2011 Workshops*. 2011, S. 60–67.

[175] **You, Y., Zhang, Z., Hsieh, C., Demmel, J. und Keutzer, K.** *100-Epoch ImageNet Training with AlexNet in 24 Minutes*. In: *arXiv preprint arXiv:1709.05011* (2017).

[176] **Zhang, H., Cisse, M., Dauphin, Y. N. und Lopez-Paz, D.** *Mixup: Beyond Empirical Risk Minimization*. In: *arXiv preprint arXiv:1710.09412* (2017).

[177] **Zhang, Z.** *A Flexible New Technique for Camera Calibration*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), S. 1330–1334.

[178] **Zhang, Z., Liu, Q. und Wang, Y.** *Road Extraction by Deep Residual U-Net*. In: *IEEE Geoscience and Remote Sensing Letters* 15.5 (2018), S. 749–753.

[179] **Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N. und Liang, J.** *UNet++: A Nested U-Net Architecture for Medical Image Segmentation*. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Hrsg. von **Stoyanov, D. et al.** Cham: Springer International Publishing, 2018, S. 3–11.

[180] **Zoph, B. und Le, Q.** *Neural Architecture Search with Reinforcement Learning*. In: *arXiv preprint arXiv:1611.01578* (2016).

[181] **Zou, W. W. und Yuen, P**. *Very Low Resolution Face Recognition Problem*. In: *IEEE Transactions on Image Processing* 21.1 (2012), S. 327–340.

Eigene Veröffentlichungen

[182] **Kaiser, C., Mitschke, N. und Dostert, K.** *Cyclic bit loading for adaptive OFDM in narrowband power line communications*. Englisch. In: *2018 IEEE International Symposium on Power Line Communications and its Applications, ISPLC 2018; Manchester; United Kingdom; 8 April 2018 through 11 April 2018*. Institute of Electrical and Electronics Engineers (IEEE), 2018.

[183] **Kaiser, J.-P., Mitschke, N., Stricker, N., Heizmann, M. und Lanza, G.** *Konzept einer automatisierten und modularen Befundungsstation in der wandlungsfähigen Produktion*. In: *Zeitschrift für wirtschaftlichen Fabrikbetrieb* 116.5 (2021), S. 313–317.

[184] **Mitschke, N. und Heizmann, M.** *Über die Detektierbarkeit von Objekten in Bildern mittels quantisierter neuronaler Netze*. In: *Technisches Messen* 86.11 (2019), S. 651–660.

[185] **Mitschke, N. und Heizmann, M.** *Advanced Quantization Methods for CNNs*. In: *2020 International Symposium on Electronics and Telecommunications (ISETC)*. 2020, S. 1–4.

[186] **Mitschke, N. und Heizmann, M.** *Semantische Segmentierung von Ankerkomponenten von Elektromotoren*. In: *Forum Bildverarbeitung 2020*. 2020, S. 329–340.

[187] **Mitschke, N. und Heizmann, M.** *Image-Based Visual Servoing of Rotationally Invariant Objects Using a U-Net Prediction*. In: *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*. IEEE. 2021, S. 235–240.

[188] **Mitschke, N., Ji, Y. und Heizmann, M.** *Task Specific Image Enhancement for Improving the Accuracy of CNNs*. In: *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM*, INSTICC. SciTePress, 2021, S. 174–181.

[189] **Mitschke, N. und M., Heizmann**. *Gerät für eine lageunabhängige optische Oberflächeninspektion*. Patentantrag DE 102021002363.3, in Prüfung. 2021.

[190] **Mitschke, N., Heizmann, M., Noffz, K.-H. und Wittmann, R.** *Ein evolutionärer Ansatz für die automatische Ermittlung der Topologie neuronaler Netze*. In: *Forum Bildverarbeitung 2018*. 2018, S. 185–196.

[191] **Mitschke, N., Heizmann, M., Noffz, K.-H. und Wittmann, R.** *Gradient Based Evolution to Optimize the Structure of Convolutional Neural Networks*. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, S. 3438–3442.

[192] **Mitschke, N., Heizmann, M., Noffz, K.-H. und Wittmann, R.** *A Fixed-Point Quantization Technique for Convolutional Neural Networks Based on Weight Scaling*. In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, S. 3836–3840.

[193] **Mitschke, N., Heizmann, M., Noffz, K.-H. und Wittmann, R.** *Ein evolutionärer Ansatz für aufgabenspezifische MobileNet-Topologien*. In: *Technisches Messen* 86.7-8 (2019), S. 413–421.

Betreute studentische Arbeiten

[194] **Aziz, R.** *Visualisierung von gelernten Merkmalen künstlicher neuronaler Netze in industriellen Anwendungen*. IIIT, 2018.

[195] **Bielski, P.** *Impact of Data Augmentation and Image Enhancement on Small ConvNets*. IIIT, 2019.

[196] **Guo, S.** *Modeling of Manufacturing Defects for the Use in Machine Learning*. IAI, IIIT, 2019.

[197] **He, C.** *Improvement of a Weakly-Supervised Classifier*. IIIT, IAI, 2019.

[198] **Huang, J.** *Automatisierter Entwurf eines künstlichen neuronalen Netzes für industrielle Anwendungen*. IAI, IIIT, 2018.

[199] **Ji, Y.** *Analysis of Preprocessing Methods in the Context of Deep Learning*. IIIT, 2020.

[200] **Liu, X.** *Transfer of CNN-Features in the Context of Industrial Image Processing*. IAI, IIIT, 2018.

[201] **Ma, H.** *Segmentation of Industrial Images Using CNNs*. IIIT, 2019.

[202] **Wei, W.** *Optimization of a CNN Implementation on a FPGA Based on MobileNet*. IIIT, 2019.

[203] **Wu, Z.** *Algorithm for an Automatic Search of Augmentation Strategies*. IIIT, 2021.

[204] **Xu, Y.** *Visualization of the Loss Function of Neural Networks*. IIIT, 2020.

[205] **Zhang, L.** *Anomalieerkennung im Bereich der industriellen Bildverarbeitung*. IIIT, 2020.

[206] **Zhang, W.** *Enlargement of a Database by Generative Adversarial Networks*. IIIT, 2018.

[207] **Zou, Y.** *Distillation and Reduction of CNNs*. IIIT, 2019.

Forschungsberichte aus der Industriellen Informationstechnik (ISSN 2190-6629)

Band 1 Pérez Grassi, Ana
Variable illumination and invariant features for detecting and classifying varnish defects.
ISBN 978-3-86644-537-6

Band 2 Christ, Konrad
Kalibrierung von Magnet-Injektoren für Benzin-Direkteinspritzsysteme mittels Körperschall.
ISBN 978-3-86644-718-9

Band 3 Sandmair, Andreas
Konzepte zur Trennung von Sprachsignalen in unterbestimmten Szenarien.
ISBN 978-3-86644-744-8

Band 4 Bauer, Michael
Vergleich von Mehrträger-Übertragungsverfahren und Entwurfskriterien für neuartige Powerline-Kommunikationssysteme zur Realisierung von Smart Grids.
ISBN 978-3-86644-779-0

Band 5 Kruse, Marco
Mehrobjekt-Zustandsschätzung mit verteilten Sensorträgern am Beispiel der Umfeldwahrnehmung im Straßenverkehr.
ISBN 978-3-86644-982-4

Band 6 Dudeck, Sven
Kamerabasierte In-situ-Überwachung gepulster Laserschweißprozesse.
ISBN 978-3-7315-0019-3

Band 7 Liu, Wenqing
Emulation of Narrowband Powerline Data Transmission Channels and Evaluation of PLC Systems.
ISBN 978-3-7315-0071-1

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 8 Otto, Carola
Fusion of Data from Heterogeneous Sensors with Distributed Fields of View and Situation Evaluation for Advanced Driver Assistance Systems.
ISBN 978-3-7315-0073-5

Band 9 Wang, Limeng
Image Analysis and Evaluation of Cylinder Bore Surfaces in Micrographs.
ISBN 978-3-7315-0239-5

Band 10 Michelsburg, Matthias
Materialklassifikation in optischen Inspektionssystemen mithilfe hyperspektraler Daten.
ISBN 978-3-7315-0273-9

Band 11 Pallauf, Johannes
Objektsensitive Verfolgung und Klassifikation von Fußgängern mit verteilten Multi-Sensor-Trägern.
ISBN 978-3-7315-0529-7

Band 12 Sigle, Martin
Robuste Schmalband-Powerline-Kommunikation für Niederspannungsverteilernetze.
ISBN 978-3-7315-0539-6

Band 13 Opalko, Oliver
Powerline-Kommunikation für Batteriemanagement-Systeme in Elektro- und Hybridfahrzeugen.
ISBN 978-3-7315-0647-8

Band 14 Han, Bin
Characterization and Emulation of Low-Voltage Power Line Channels for Narrowband and Broadband Communication.
ISBN 978-3-7315-0654-6

Band 15 Alonso, Damián Ezequiel
Wireless Data Transmission for the Battery Management System of Electric and Hybrid Vehicles.
ISBN 978-3-7315-0670-6

Band 16 Hernández Mesa, Pilar
Design and analysis of a content-based image retrieval system.
ISBN 978-3-7315-0692-8

Band 17 Suchaneck, André
Energiemanagement-Strategien für batterieelektrische Fahrzeuge.
ISBN 978-3-7315-0773-4

Band 18 Bauer, Sebastian
Hyperspectral Image Unmixing Incorporating Adjacency Information.
ISBN 978-3-7315-0788-8

Band 19 Vater, Sebastian
Monokulare Blickrichtungsschätzung zur berührungslosen Mensch-Maschine-Interaktion.
ISBN 978-3-7315-0789-5

Band 20 Back, Kristine
Erkennung menschlicher Aktivitäten durch Erfassung und Analyse von Bewegungstrajektorien.
ISBN 978-3-7315-0909-7

Band 21 Nürnberg, Thomas
Entwurf von Computational-Imaging-Systemen am Beispiel der monokularen Tiefenschätzung.
ISBN 978-3-7315-0941-7

Band 22 Kaiser, Cornelius
Adaptive Modulationsverfahren für die schmalbandige Powerline-Kommunikation in Niederspannungsnetzen.
ISBN 978-3-7315-1010-9

Band 23 Struckmeier, Frederick
Prozesssicherheit von Laserschneidmaschinen – Auflagemessung und Schachtelung
ISBN 978-3-7315-1127-4

Band 24 Tatzel, Leonie Felica
Verbesserungen beim Laserschneiden mit Methoden des maschinellen Lernens
ISBN 978-3-7315-1128-1

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 25 Wetzel, Johannes
Probabilistic Models and Inference for Multi-View People Detection in Overlapping Depth Images
ISBN 978-3-7315-1177-9

Band 26 Mitschke, Norbert
Konvolutionäre neuronale Netze in der industriellen Bildverarbeitung und Robotik
ISBN 978-3-7315-1197-7



ISSN 2190-6629
ISBN 978-3-7315-1197-7

Gedruckt auf FSC-zertifiziertem Papier

